Free University of Bozen • Bolzano

Faculty of Computer Science

A Software Assurance Model for Mobile Applications

Luis Ricardo CORRAL VELÁZQUEZ

A Dissertation Submitted in

Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy

Thesis Supervisor:

Prof. Ing. Giancarlo Succi

Thesis Co-Supervisor:

Prof. Arrigo Luigi Frisiani

Relatori – Doktorväter – Advisors:

Prof. Ing. Giancarlo Succi

Prof. Arrigo Luigi Frisiani

Commisione Giudicatrice – Prüfungskommission – Examination Committee:

Prof. Ing. Luigi Benedicenti, University of Regina, Canada

Prof. Corrado Aaron Visaggio, Università del Sannio, Benevento, Italy

Prof. Ing. Alberto Sillitti, Free University of Bozen · Bolzano, Italy

Data della discussione publica – Datum der öffentliche Doktoratsprüfung – Public defense held on:

March 25th, 2014

Keywords:

Application, GQM, ISO 25010, Mobile, Product, Quality, Standard

 Luis Ricardo Corral Velázquez
Center for Applied Software Engineering
Faculty of Computer Science
Free University of Bozen · Bolzano

Piazza Domenicani 3 Dominikanerplatz
39100 Bolzano-Bozen, Italy

luis.corral@unibz.it
+39 0471 016245
www.inf.unibz.it/~lcorralvelazquez

This Thesis is dedicated to my wife, parents, siblings and to the loving memory of my late grandparents.

ABSTRACT

Mobile applications have become ubiquitous, adopted by millions of users that register billions of downloads a day. To increase the competitiveness of the mobile software product, developers should care in a very detailed fashion about the qualities demanded by end users, execution targets and mobile markets. The quality of mobile applications is controlled by market policies, and is judged by customer's reviews and ratings. Software developers need to count on reference points to assess the product against expected characteristics in an objective fashion, without the need of completing the customer feedback cycle.

This work contributes on the configuration of a software assurance model that learns from well-founded quality methodologies adapting the necessary characteristics to meet the requirements of mobile software applications. We extracted the most relevant quality requirements from the mobile application market and we generated quality instruments that help developers to produce applications that meet and exceed such requirements. We implemented a Quality Function Deployment that allowed us to transform the requirements of the mobile application market into quality characteristics that can be assured in the software development cycle. In particular, we introduced the Mobile App Quality Model, that leverages the ISO/IEC 25010 quality standard as a mechanism to assure the fulfillment of the quality requirements from the mobile application stores, based on standard quality characteristics. The Mobile App Quality Model and the associated techniques presented in this thesis aim to provide the mechanisms to assess the quality of the mobile software product built upon mobile-specific, market-aware requirements. We deployed a number of case studies and surveys on custom applications and real-world products, which let us understand the actual impact of several of the quality characteristics identified by the Mobile App Quality Model, illustrating its usefulness. The case studies and surveys also served as example to exercise the different quality tools developed in this thesis for the quantitative analysis of the software product utilizing software metrics.

ACKNOWLEDGEMENTS

The author gratefully acknowledges the support of:

- Professor Giancarlo Succi, who trusted me and accepted to be my supervisor during this three years. His advice and leadership are reflected in this Thesis.
- Professor Alberto Sillitti, whose untiring help, guidance and patience were instrumental on taking this work to the end.
- Professor Arrigo Luigi Firisiani, for his detailed reviews, advise and genuine interest on making this Thesis a high quality piece of research.
- The staff of the Faculty of Computer Science: The Research, Teaching and Administrative bodies gave all the support and resources to carry out this work. In particular, the Offices of the Dean, the Director of the PhD programme, and Secretariat. Thank you Pekka, Carmen, Claudia, Federica, Ines, Viviana.
- The amazing staff of the Center for Applied Software Engineering: Andrea, Andrej, Anton, Barbara, Bruno, Etiel, Ines, Marko, Nabil, Saulius: Thank to you all for your support and cheerfulness. Thank you Ilenia for all your help.
- To my PhD fellows and the Academic Staff of the Faculty of Computer Science. I am honored to work in a world-class team. Thank you for your advice and example. I am specially grateful to the professors I had the pleasure to help as assistant, and to the students I had the honor to serve as instructor.
- To our witnesses Adriano and Sara, Anton and Tanya, and to all our friends in Italy who supported us at all times. To list you all I would need a whole Thesis.
- Un especial agradecimiento a mi familia y amigos en México por su apoyo, comprensión, y sobre todo por su paciencia. Padres, hermanos, abuelita, tíos, oscuros, amigotes: La lejanía ha valido la pena.
- Of course the most important thanks go to Naomi, for making me believe that I was able and willing to carry out this endeavor. I left everything I had for this.
- *Deo Gratias.*

This work would not have been possible without the PhD scholarship generously granted by the Free University of Bozen · Bolzano.

Selected excerpts of the research works presented in this Thesis were supported by the European Social Fund "EN-ACT" Interreg IV Project Grant.

FOREWORD

Many years ago, my mother taught me how to read. Another day, my father taught me how to understand. A number of teachers showed me how to learn. I owe them all what I know and all what I am.

In Mexico I read books and heard the stories of magnificent professors who, back in the Middle Ages, discussed about the essence of things. I was amazed to read about the Problem of the Universals, and I was thrilled to see how this dispute of the XII century relates to our current Computer Science notion of classes and objects. What was the mysterious charisma of the Scholastic Doctors that seduced me to the point to wish to become one of them?

I dreamed about the lectures of John Duns Scotus, Peter Abelard, William of Ockham, Saint Bonaventure or Saint Thomas Aquinas. Some of them wrote the statutes of the old Studia Generalia, which evolved into the current European Universities. Since always, I wanted to be part of them. "You should go to the sources", said a voice of wisdom. So I did it.

My blessed Grandfather wrote a Bachelor Thesis. My beloved Father wrote a Master Thesis. Now I set forth this Doctoral Thesis. This document sums up 33 years of continuous learning. I completed the *trivium* and *quadrivium.*

Affreschi della Chiesa dei Francescani, Bolzano (XV Secolo)

TABLE OF CONTENTS

CHAPTER 1:

INTRODUCTION....1

1.1.Mobile software systems....2

1.2.Object of study: Mobile Software Engineering....4

1.3.Methodology....7

1.4.Thesis contribution....8

1.5.Structure of the Thesis....9

1.6.Scientific work products....10

1.7.Chapter bibliography....12

CHAPTER 2:

QUALITY ASSURANCE FOR MOBILE SOFTWARE: RESEARCH ASPECTS....15

2.1.Research statement....17

2.2.Research questions....19

2.3.Research strategy....20

2.4.Standard-based definition of quality characteristics....21

2.5.Research forecast....22

2.6.Chapter bibliography....23

CHAPTER 3:

SOFTWARE ASSURANCE FOR MOBILE PRODUCTS: STATE OF THE ART....25

3.1.State of the Art and the Practice: Research questions....25

3.2.Literature review strategy....26

3.3.Review of software practices for mobile software....27

3.3.1.Process oriented practices....28

3.3.2.Product oriented practices....32

3.3.3.Implementation oriented practices....33

3.4.Literature review results....34

3.4.1.Suitability....34

3.4.2.Evidence....36

3.4.3.Evolution....39

3.5.Open items....41

3.6.Summary and conclusions....43

3.7.Chapter bibliography....44

CHAPTER 4:
QUALITY ATTRIBUTES OF MOBILE APPLICATIONS....51
4.1.Application stores as software delivery platform....53
4.1.1.The App Store business model....53
4.1.2.Software quality assurance in mobile application markets....56
4.2.Software quality requirements from mobile application stores....58
4.2.1.Extracting quality requirements from mobile application stores....60
4.2.2.Comparative analysis....64
4.3.Discussion....66
4.4.Chapter bibliography....67
CHAPTER 5:
DEFINING MOBILE SPECIFIC QUALITY ATTRIBUTES....71
5.1.Defining metrics after standardized quality attributes....71
5.1.1.Internal and External quality: Developer's point of view....73
5.1.2.Quality in Use: Customer's point of view....77
5.2.Associating ISO/IEC 25010 with mobile-specific quality requirements....80
5.2.1.Implementation of the Quality Function Deployment....81
5.2.2.Prioritization of the quality characteristics....87
5.3.Composition of the market-based Mobile App Quality Model....89
5.3.1.Mobile Internal and External quality: Developer's point of view....92
5.3.2.Quality in Use: Customer's point of view....95
5.4.Creating mobile-specific quality metrics with a partially-instantiated GQM....97
5.5.Chapter bibliography....102
CHAPTER 6:
MEASURING PRODUCT QUALITY OF CUSTOM APPLICATIONS....105
6.1.Case study 1: Measuring performance of a custom application....105
6.1.1.Implementation of the partially-instantiated GQM....106
6.1.2.Experimental setting....107
6.1.3.Methodology....108
6.1.4.Mobile application....108
6.1.5.Data analysis....109
6.1.6.Discussion....110
6.1.7.Case study conclusions....112
6.2.Case study 2: Measuring energy consumption of a custom application....113
6.2.1.Implementation of the partially-instantiated GQM....114
6.2.2.Data sources....115

6.2.3.Mobile application....115
6.2.4.Experimental setting....118
6.2.5.Data analysis....119
6.2.6.Case study conclusions....121
6.3.Summary and conclusions....122
6.4.Chapter bibliography....123

CHAPTER 7:
MEASURING PRODUCT QUALITY IN REAL MARKET APPLICATIONS....127
7.1.Product evaluation and market success....128
7.2.Relating mobile code quality and market success....129
7.2.1.Implementation of the partially-instantiated GQM....130
7.2.2.Selected application market....134
7.2.3.Study of code quality and market success on open source Android applications....135
7.2.4.Data collection plan....137
7.2.5.Data analysis....139
7.2.6.Understanding the impact of product quality in app store success....146
7.2.7.Discussion....152
7.3.Summary and conclusions....155
7.4.Chapter bibliography....155

CHAPTER 8:
SUMMARY AND CONCLUSIONS....159
8.1.Research questions revisited....160
8.2.Experiments, case studies and surveys....162
8.3.Limitations and threats to validity....163
8.4.Directions of future research....163
8.5.Closing remarks....164

INDEX OF TABLES

Table 1. Summary of research works presenting mobile software development practices....28
Table 2. Mobile software assurance practices and their implementations....38
Table 3. Mapping of Agile ground themes and traits on mobile software development....40
Table 4. Quality requirements in major application stores....62
Table 5. Summary and frequency of quality characteristics for major mobile application stores....64
Table 6. Demanded quality weight and demanded quality relative weight....82
Table 7. Internal/External quality sub-characteristic weight and relative weight....85
Table 8. Quality in Use sub-characteristic weight and relative weight....86
Table 9. Mobile App Quality Model: selected Internal/External quality sub-characteristics....90
Table 10. Mobile App Quality Model: selected Quality in Use sub-characteristics....91
Table 11. Evaluation of execution time of Android native and web application....110
Table 12. Energy consumption stressing routines....119
Table 13. Percentage of battery discharge in two hours....120
Table 14. Ranking of energy hungry components....120
Table 15. Descriptive statistics for product oriented metrics of 100 Google Play applications....139
Table 16. Descriptive statistics for market oriented metrics of 100 Google Play applications....141
Table 17. Non-parametric rank correlation (Spearman) of product oriented metrics....143
Table 18. Non-parametric correlation (Spearman) of market oriented metrics....144
Table 19. Descriptive statistics for the success index of 100 Google Play applications....145
Table 20. Non-parametric correlation (Spearman) between product oriented metrics and success index....147
Table 21. Multiple regression model: product complexity and market success index....150
Table 22. Regression model: product coupling and market success index....151
Table 23. Regression model: product cohesion and market success index....151
Table 24. Regression model: product size and market success index....152
Table 25. Mobile App Quality Model: code-relevant quality sub-characteristics....153

Index Of Figures

Figure 1. Available apps on iOS Store, Google Play and Windows Phone Store....3
Figure 2. Number of downloads on iOS Store and Google Play....3
Figure 3. Average developer revenue in diverse application targets....3
Figure 4. High-level quality drivers of mobile software applications....17
Figure 5. Multi-dimensional approach to measure the quality of mobile software applications....18
Figure 6. Approach for approximating the quality of the mobile software product....22
Figure 7: Processes, product assurance practices and implementation practices....36
Figure 8: Software product quality reference model from ISO/IEC 25010 72
Figure 9: ISO/IEC 25010 product quality model: Internal and External quality characteristics....73
Figure 10: ISO/IEC 25010 product quality model: Quality in Use characteristics....78
Figure 11: Internal and External quality sub-characteristics ranked by relative weight....88
Figure 12: Quality in Use sub-characteristics ranked by relative weight....89
Figure 13: Mobile App Quality Model: development-oriented quality characteristics....92
Figure 14: Mobile App Quality Model: user-oriented quality characteristics....95
Figure 15: Operational definition of the measured job....107
Figure 16: Android native application and PhoneGap web application....109
Figure 17: PhoneGap's method call flow path....111
Figure 18: Android application architecture....116
Figure 19: Product quality model based on source code metrics....133
Figure 20: Selected market success model based on app store metrics....134
Figure 21: High-level data collection plan for source code metrics and app store metrics....138
Figure 22: Data distribution of source code metrics 140
Figure 23: Data distribution of app market metrics....142
Figure 24: Data distribution of the success index metric....145

Chapter 1: Introduction

The impact of the mobile software product is growing every day, reaching a point in which mobile devices have become one of the most important platforms for the distribution and utilization of software. Smartphone sales outnumber those of PCs (IDC 2012a, Canalys 2012), and application markets represent a primary channel for the dissemination of end-user software products, hosting thousands of apps and reporting millions of downloads per day (IDC 2012b). Handset terminals have experienced a shift from being simple communication devices to become high-end, multipurpose computer equipment. Smartphones are driven by powerful operating systems that allow users to add and remove applications, employing an architecture that is similar to a regular personal computer. However, these applications have to cope with several constraints inherent to the mobile ecosystem that are not present in conventional desktop computing; these constraints include the limited capabilities of terminal devices, wireless communication problems, mobility issues, the diversity of standards and operating platforms, security matters, and many others.

The quality of mobile applications is usually regulated by application market policies, and the user's perception of the quality is communicated by "word of mouth" via customer's ratings and reviews. The high relevance gained by mobile software applications and the high competitiveness of the mobile market trigger the need for methods to measure the quality of mobile software products from a target-specific point of view. In an environment full of constraints, successful software should comply with clear development practices that involve different subjects like algorithms,

systems architecture, communications, human-machine interface and other disciplines. A solid understanding of them is of paramount importance to broad the scope of Software Engineering practices to conduct mobile software development projects too.

1.1 Mobile software systems

Mobile systems carry out their operations in a wide range of environments, from traditional services like voice communication, text messaging and games to business applications, location based services, augmented reality, productivity tools and much more. Although software has played a major role as operative platform for portable communication devices, in the last 10 years it has grown in importance, evolving to become an important component of user space. Today, mobile systems challenge desktop systems as the most important platform to distribute and utilize software, and as consequence they become a very attractive target to develop for.

Mobile application markets host thousands of products, and downloads reach numbers in the order of millions a day. "App stores" grow as primary and influential channel for the dissemination of end-user mobile software products. As of 2013, application distribution markets cover a range of more than 800,000 applications in the Android App Market Store and more than 750,000 in the iOS App Market (Figure 1), involving a cumulative number of 50,000,000,000 downloads from the iOS App Store, followed closely by the Android App Market (Figure 2).

As mobile applications are one of today's fastest growing digital sectors, developers struggle finding ways to profit from for their work: competition is big, and the mobile software product is usually low-priced; nonetheless, in spite of this, it is a field that has proved to generate important revenues and it has become very attractive for software developers (Figure 3). Applications, thus, have to be created and monetized incorporating the appropriate strategies and methodologies that comprise technical, marketing and business aspects.

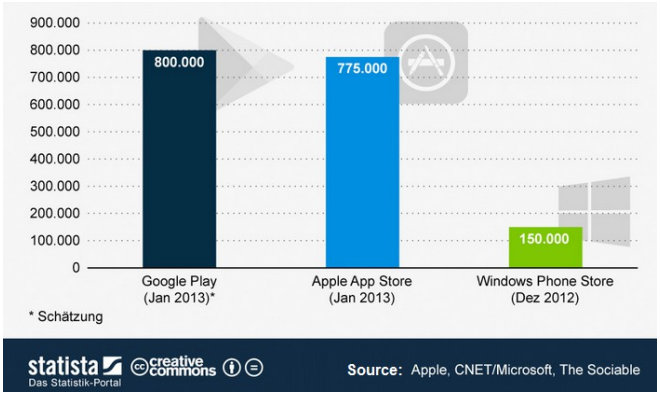


Figure 1. Available apps on iOS Store, Google Play and Windows Phone Store[1]

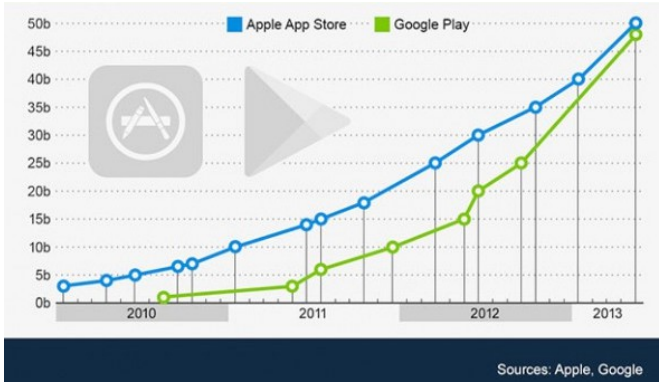


Figure 2. Number of downloads on iOS Store and Google Play[2]

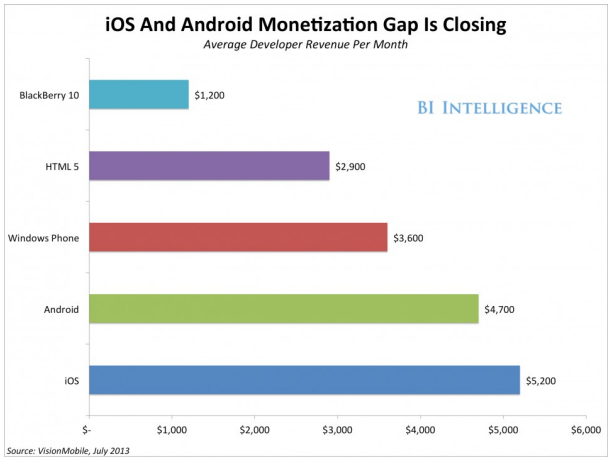


Figure 3. Average developer revenue in diverse application targets[3]

[1] http://www.statista.com

[2] http://www.statista.com

[3] http://www.businessinsider.com

The increasing capabilities of mobile devices and their adoption by a vast range of markets attracts the interest of knowing in a more detailed fashion the conditions under which mobile software is conceived, designed, implemented, maintained and measured, both from process and product point of views.

Software applications for mobile devices are developed through practices whose objective is to create products that are able to perform satisfactorily in a limited environment that, still, puts high requirements on availability, efficiency and responsiveness. Mobile software developers must bear in mind such requirements, along with the regulations imposed by distribution channels, which must be met in order to promote a product in any application store. This poses on developers the question of how to produce successful apps in a restrictive environment with a wide and competitive market. Mobile software engineers should put in practice a development strategy that considers several quality drivers: the mobile environment itself, the expectations of the end user, and the restrictions set by the mobile application market.

1.2 Object of study: Mobile Software Engineering

Since the early 2000s, it has been remarked that the development of mobile applications is particularly challenging due to the specific demands and technical constraints of the mobile environment (Abrahamsson 2004). It was noted as well that the mobile business model differs from the general desktop software business model (Yamakami 2005, Yamakami 2008). Since then, several software life cycles have been proposed to assist developers in the process of creating suitable mobile applications able to succeed in the mobile market. Nonetheless, the proposed software life cycles tend to fail in establishing a clear link between the general quality goals of the mobile applications and the specific characteristics that should be measured to determine the fulfillment of such goals. This situation makes it difficult to choose a development model that leads with confidence to a dependable, high quality final product (Glissmann 2005).

Software Engineering focuses on the study of software artifacts, people as producers of software, and processes as guarantors of quality. Roman et al. (2000) related for the

first time Software Engineering and mobile platforms by providing a research agenda that includes theoretical investigation, models, algorithms, applications, system research and middleware, inviting the Software Engineering community to embrace mobile computing as the new frontier to conquer. In consequence, "Mobile" Software Engineering should undertake scientific methodologies for understanding and applying Software Engineering principles to the development of software for mobile devices, including software development processes, system architectures, system performance, software measurement, and software quality and improvement.

Wasserman (2010) noted a lack of research on Software Engineering processes that fit into mobile software development, urging to understand and study the particularities given by a mobile environment throughout the whole software development process. Moreover, he noted as well that the mobile environment leaves behind the traditional software-centric perspective to incorporate a number of external factors to consider to design and implement an application of this kind.

The development of mobile applications has been be defined as a very challenging task due to the specific demands and technical constraints of the environment, in which we may point out as critical factors (Abrahamsson 2005):

- the limited capabilities and rapid evolution of terminal devices,
- the incorporation of various development standards,
- the need of dealing with diverse protocols and network technologies,
- the need of operating on a variety of platforms, addressing specific needs of mobile terminal users, and
- the need of meeting strict time-to-market requirements.

Spataru (2010) defined a summary and categorization of the limitations of mobile computing devices previously outlined by Hayes (2003). Such categorization considers two types of constraints: evolving and inherent:

- evolving constraints: they include current limitations that will be solved in the future by the evolution in technology and resources, bandwidth, signal coverage, etc. For instance, dealing with a slow data network.
- inherent constraints: they are those intrinsic to mobile platforms and target devices as they are by design; this family of limitations is permanent and will not be solved in the near future, since it is part of the native characteristics of the mobile execution environment. For example, interfacing with the phone via a limited keyboard or tapping a small screen.

This simple taxonomy gains in importance when software designers define approaches and practices that aim to relieve feasible constraints by means of non-functional requirements.

Specialized literature points out other important issues for mobile software like the variety of wireless communication problems (availability, bandwidth variability, heterogeneous networks), assorted mobility issues (migration, management of location-dependent information), the variety of standards, protocols and network technologies, the limited capabilities of terminal devices (low power supplies, small sized user interfaces, and low storage capacities), privacy, etc. (Rahimian 2008).

Other notorious concerns include the growing need of targeting more than one platform (e.g., iOS, Android, Windows Mobile, and others) to widen the range of potential customers, since each one has a high number of users representing potential customers of a new application (Mikkonen 2011, Taivalsaari 2011). In this aspect, from a business perspective, we should mention that the production and marketing of mobile software products should satisfy a business model able to generate revenues in a highly competitive market, given that mobile applications tend to be low priced. Developers would not want to dismiss an extensive spectrum of users by developing only for one platform, but at the same time conducting the whole development process for a single application for each platform eventually becomes redundant and expensive.

These factors illustrate only some of the challenges that software engineers should consider to deliver a functional, error-free software product that is suitable to run on a

limited platform, and that effectively meets the mobile market's needs to generate the expected revenues.

Software engineers count on a rich variety of software processes, methodologies and quality management systems intended to support the development of general-purpose software products. In an attempt to cope with the challenges of the mobile development, a more limited number of mobile-specific development models have been introduced. Such efforts have in common the selection of Agile methodologies to provide a development framework. Commonly, mobile-specific development models underline the necessity of adapting software development practices to the evolving needs of mobile software, but they pay little attention to software assurance tasks due to the traditionally low level of criticality of mobile applications (Abrahamsson 2003). In consequence, currently it is not available a software paradigm to assure the quality of the mobile software product, taking into account the needs and particularities of the mobile environment and the mobile application market.

As software for mobile devices earns a key role in the operation and utilization of mobile devices, and the utilization of mobile apps systematically grows, the lack of a software assurance model for mobile applications is more noticeable. A solid knowledge of software quality assurance aspects may enable stakeholders to make decisions that impact positively the whole software development process and allow to deliver better mobile applications (Shiratuddin 2008).

1.3 Methodology

To determine what are the characteristics, gaps and differences of the current software development processes with respect to the mobile-specific quality assurance needs, a Literature Survey was conducted. After completion of this phase, we outlined the work hypothesis and associated research questions. We utilized a suitable quality methodology to provide a solid approach to answer our research questions. Finally, we conducted a series of experiments and surveys to implement our approach on custom and real-world mobile software products.

We carried out a thorough theoretical and practical analysis that allowed us to identify the strengths and development needs of the current State of the Art and Practice with respect to what is required by the mobile software process and product. In addition, this thesis includes a field study conducted in a real-world setting, that permitted us to appraise the applicability and results of our methodology. Additional information regarding research aspects is described comprehensively in Chapter 2 of this document.

1.4 Thesis contribution

We believe that it is necessary to study the software quality requirements of the mobile software product to establish an assurance framework that helps developers to produce high quality mobile applications. This work contributes on the configuration of a robust software assurance model that learns from traditional and innovative methodologies that have proven their value in the development of conventional software, adapting the necessary characteristics to meet the needs of mobile applications.

Through this work, different software development processes and quality assurance methodologies will be analyzed and implemented to identify and address the mobile-specific assurance needs. For this reason, it will be necessary to cultivate a deep understanding of the mobile-specific software development processes and about the requirements and constraints given by the mobile execution environment, end users, and application markets.

The objective of this thesis is to provide researchers and practitioners with instruments to manage and assure a software development life cycle in which the final product is targeted to run in a mobile device such as a cellular phone or a tablet. Through theoretical foundations and practical experimentation, this research sets the ground for software engineers to understand how to project and conduct a software development processes able to produce high-quality mobile software applications.

1.5 Structure of the Thesis

This dissertation is organized as follows:

- Chapter 1: introduces the topic of Mobile Software Engineering, establishes the motivation of this work, and its scientific contribution;
- Chapter 2: presents the research approach, including the work hypothesis and research questions;
- Chapter 3: covers the State of the Art on Mobile Software Engineering, analyzes the most important research works on the subject matter;
- Chapter 4: defines the quality assurance needs on the mobile app ecosystem, in the context of the quality considerations of major mobile application stores;
- Chapter 5: conducts the selection of the mobile-specific quality characteristics, and structures them after the ISO/IEC 25010 software quality standard implementing a Quality Function Deployment;
- Chapter 6: defines metrics through a partially instantiated GQM and exercises them through case studies that measure a subset of relevant quality metrics on custom mobile applications;
- Chapter 7: defines metrics through a partially instantiated GQM and exercises them in a survey that measures a set of general-purpose quality metrics on applications taken from a real world mobile app store;
- Chapter 8: provides directions for future research, summarizes the topics of our discussion and draws conclusions.

In summary: Chapters 1 and 2 contain the research aspects of the underlying work; Chapter 3 reports the Literature Review; Chapters 4, 5, 6 and 7 present the research carried out, and Chapter 8 summarizes the work and draws conclusions.

1.6 Scientific work products

Selected excerpts of this thesis were presented in a number of international refereed venues, in the form of 9 conference papers and 4 workshop papers. We include the Chapter number that concentrates the majority of the contents of each paper.

2014

1. **Corral, L.**; Georgiev, A.B.; Sillitti, A.; Succi, G.; (2014) Method reallocation to reduce energy consumption: An implementation in Android OS. Accepted for Publication as a full paper in the *29th ACM Symposium on Applied Computing (SAC 2014)* (Chapter 6).

2013

2. **Corral, L.**; Sillitti, A.; & Succi, G.; (2013) Agile software development processes for mobile systems: Accomplishment, evidence and evolution. In *Proceedings of the 10th International Conference on Mobile Web Information Systems (MobiWIS 2013).* Lecture Notes in Computer Science, vol. 8093. pp. 90-106. Springer-Verlag Berlin / Heidelberg. ISBN: 978-3-642-40275-3. doi:10.1007/978-3-642-40276-0_8 (Chapter 3)

3. **Corral, L.**; Sillitti, A.; & Succi, G.; (2013) Using a partially instantiated GQM to measure the quality of mobile apps. In *Proceedings of the 25th International Conference on Software Engineering and Knowledge Engineering (SEKE 2013).* pp. 520-524. Knowledge Systems Institute. ISBN: 978-1-891706-33-2. (Chapter 5)

4. **Corral, L.**; Georgiev A.B.; Sillitti, A.; & Succi, G.; (2013) A method for characterizing energy consumption in Android smartphones and tablets. In *Proceedings of the 2nd International Workshop on Green and Sustainable Software (GREENS 2013), in connection with ICSE 2013.* pp. 38-45. IEEE. ISBN: 978-1-4673-6267-2. doi:10.1109/GREENS.2013.6606420 (Chapter 6)

5. **Corral, L.**; Sillitti, A.; & Succi, G.; (2013) Software development processes for mobile systems: Is Agile really taking over the business? In *Proceedings of the 1st International Workshop on Mobile-Enabled Systems (MOBS 2013), in*

connection with ICSE 2013. pp. 19-24. IEEE. ISBN: 978-1-4673-6333-4. doi:10.1109/MOBS.2013.6614218 (Chapter 3).

2012

6. **Corral, L.**; (2012) Standard-based strategy to assure the quality of the mobile software product. In *Proceedings of the 3rd annual conference on systems, programming, and applications: software for humanity (SPLASH 2012).* pp. 95-96. ACM. ISBN: 978-1-4503-1563-0. doi:10.1145/2384716.2384755 (Chapter 2).
7. **Corral, L.**; (2012) Using software quality standards to assure the quality of the mobile software product. In *Proceedings of the 3rd annual conference on systems, programming, and applications: software for humanity (SPLASH 2012).* pp. 37-40. ACM. ISBN: 978-1-4503-1563-0. doi:10.1145/2384716.2384734 (Chapter 2).
8. **Corral, L.**; Sillitti, A.; & Succi, G.; (2012) Mobile multiplatform development: An experiment for performance analysis. In *Proceedings of the 9th International Conference on Mobile Web Information Systems (MobiWIS 2012).* Procedia Computer Science, vol. 10. pp. 736-743. Elsevier. ISSN: 1877-0509. doi:10.1016/j.procs.2012.06.094 (Chapter 6).
9. **Corral, L.**; Janes, A.; & Remencius, T; (2012) Potential advantages and disadvantages of multiplatform development frameworks – A vision on mobile environments. In *Proceedings of the 3rd International Workshop on Service Discovery and Composition in Ubiquitous and Pervasive Environments (SUPE 2012), in connection with MobiWIS 2012.* Procedia Computer Science, vol. 10. pp. 1202-1207. Elsevier. ISSN: 1877-0509. doi:10.1016/j.procs.2012.06.173 (Chapter 6).
10. **Corral, L.**; Sillitti, A.; Succi, G.; Strumpflohner, J.; & Vlasenko, J; (2012) DroidSense: a mobile tool to analyze software development processes by measuring team proximity. In *Proceedings of the 50th international conference on Objects, Models, Components, Patterns (TOOLS'12).* Lecture Notes in Computer Science, vol. 7304. pp. 17-33. Springer-Verlag Berlin / Heidelberg. ISBN: 978-3-642-30560-3. doi:10.1007/978-3-642-30561-0_3 (Chapter 6).

2011

11. **Corral, L.**; Sillitti, A.; Succi, G.; Garibbo, A.; & Ramella, P; (2011) Evolution of mobile software development from platform-specific to web-based multiplatform paradigm. In *Proceedings of the 10th SIGPLAN symposium on new ideas, new paradigms, and reflections on programming and software (ONWARD 2011).* pp. 181-183. ACM. ISBN: 978-1-4503-0941-7. doi:10.1145/2048237.2157457 (Chapter 6)

12. **Corral, L.**; Sillitti, A.; & Succi, G.; (2011) Preparing mobile software development processes to meet mission-critical requirements. In *Proceedings of the 2nd Annual Workshop on Software Engineering for Mobile Application Development. In conjunction with MOBICASE 2011.* pp. 9-11. (Chapter 3).

13. **Corral, L.**; Sillitti, A.; & Succi, G.; (2011) Managing TETRA channel communications in Android. In *Proceedings of the 13th International Conference of Enterprise Information Systems (ICEIS 2011).* vol. 3. pp. 307-312. SciTePress. ISBN: 978-989-8425-55-3. (Chapter 3).

1.7 Chapter bibliography

(Abrahamsson 2003) Abrahamsson, P.; Warsta, J.; Siponen, M.T.; & Ronkainen, J.; (2003) New directions on agile methods: a comparative analysis. In *Proceedings of the 25th International Conference on Software Engineering 2003 (ICSE 2003).* pp. 244-254. IEEE Computer Society. doi:10.1109/ICSE.2003.1201204

(Abrahamsson 2004) Abrahamsson, P.; Hanhineva, A.; Hulkko, H.; Ihme, T.; Jäälinoja, J.; Korkala, M.; Koskela, J.; Kyllönen, P.; & Salo, O.; (2004) Mobile-D: an agile approach for mobile application development. In *Proceedings of the 19th annual ACM SIGPLAN conference on object-oriented programming systems, languages, and applications (OOPSLA '04).* pp. 174-175. ACM. doi:10.1145/1028664.1028736

(Abrahamsson 2005) Abrahamsson, P; (2005) Mobile software development – the business opportunity of today. In Proceedings of the International Conference on Software Development pp. 20-23.

(Canalys 2012) Canalys (2012) Smart phones overtake client PCs in 2011. Press Release. Available online: http://www.canalys.com/newsroom/smart-phones-overtake-client-pcs-2011 (Accessed on June 6th, 2013).

(Glissmann 2005) Glissmann, S.; Smolnik, Stefan; Schierholz, R.; Kolbe, L.; & Brenner, W.; (2005) Proposition of an m-business procedure model for the development of mobile user interfaces. In *Proceedings of the International Conference on Mobile Business, 2005 (ICMB 2005).* pp. 308-314. IEEE Computer Society. doi:10.1109/ICMB.2005.83

(Hayes 2003) Hayes, I. S.; (2003). *Just enough wireless computing.* Prentice Hall. ISBN: 978-0536750624.

(IDC 2012a) International Data Corporation (2012) *Nearly 1 billion smart devices shipped in 2011.* Press Release. Available online: http://www.idc.com/getdoc.jsp?containerId=prUS23398412 (Accessed on June 6th, 2013).

(IDC 2012b) International Data Corporation (2012) *Nearly 183 billion annual mobile app downloads by 2015.* Press Release. Available online: http://www.idc.com/getdoc.jsp?containerId=prUS23398412 (Accessed on June 7th, 2012).

(Mikkonen 2011) Mikkonen T; & Taivalsaari A.; (2011) Apps vs. open web: The battle of the decade. In *Proceedings of the 2nd Annual Workshop on Software Engineering for Mobile Application Development. In conjunction with MOBICASE 2011.* pp. 22-26.

(Rahimian 2008) Rahimian, V.; Habibi, J.; Rahimian, V.; & Habibi, J.; (2008) Performance evaluation of mobile software systems: Challenges for a software engineer. In *Proceedings of the 5th International Conference on Electrical Engineering, Computing Science and Automatic Control, 2008 (CCE 2008).* pp. 346-351. IEEE. doi:10.1109/ICEEE.2008.4723426

(Roman 2000) Roman, G.C.; Picco, G. P.; & Murphy, A.L.; (2000) Software engineering for mobility: a roadmap. In *Proceedings of the International Conference on Software Engineering (ICSE 2000).* pp. 241-258. ACM. doi:10.1145/336512.336567

(Shiratuddin 2008) Shiratuddin, N.; & Sarif, S. M. (2009). The md-Matrix: a learning tool in the mobile application development course. *International Journal of Mobile Communications.* vol. 7 (4) pp. 494-514. InderScience. doi:10.1504/IJMC.2009.023696

(Spataru 2010) Spataru, A.C.; (2010). *Agile development methods for mobile applications.* Master Thesis. School of Informatics, University of Edinburgh, United Kingdom.

(Taivalsaari 2011) Taivalsaari, A.; Mikkonen, T.; Anttonen, M.; Salminen, A.; (2011) The death of binary software: End user software moves to the web. In *Proceedings of the 2011 Ninth International Conference on Creating, Connecting and Collaborating through Computing (C5).* pp. 17-23. IEEE. doi:10.1109/C5.2011.9

(Wasserman 2010) Wasserman, A.I.; (2010) Software engineering issues for mobile application development. In *Proceedings of the FSE/SDP workshop on future of Software Engineering research (FoSER '10).* pp. 397-400. ACM. doi:10.1145/1882362.1882443

(Yamakami 2005) Yamakami, T.; (2005) Mobile application platform strategies: business model engineering for the data intensive mobile age. In *Proceedings of the 4th International Conference on Mobile Business, 2005 (ICMB '05).* pp. 333-339. IEEE Computer Society. doi:10.1109/ICMB.2005.62

(Yamakami 2008) Yamakami, T.; (2008) Business model engineering analysis on mobile client-side software platform strategies. In *Proceedings of the 7th International Conference on Mobile Business, 2008 (ICMB '08).* pp. 59-64. IEEE Computer Society. doi:10.1109/ICMB.2008.21

Chapter 2:

Quality Assurance for Mobile Software: Research Aspects

Having set as a goal the configuration of a quality assurance model for mobile applications, to predict the quality of a mobile software product we have to determine reference points to compare the product against a series of expected characteristics, in such a way that a given application can be assessed quantitatively. This is, effectively, the high level goal of the underlying work. However, defining "mobile software quality" may be a challenging task: understanding the mobile domain and its quality drivers is of vital importance.

To guarantee the success of a mobile software project based on stringent software quality assurance practices, we need to broaden our knowledge of the elements that drive the quality of the mobile software product, including market policies, physical constraints, user expectations, and others. In the same sense, we need to know the current status of the methodologies and activities concerned with guaranteeing the quality of the mobile software product; for instance, software life cycles, development methodologies, best practices, etc.

Thus, we need to establish a research agenda to investigate what are the most important quality requirements posed on mobile applications, what are their origins, relevance and impact. Then, we need to explore whether it is possible or not to

associate these quality requirements with characteristics or attributes that can be measured.

As an initial step, we propose to work with three big sources of quality requirements for mobile applications, namely end-users, execution environment and mobile application markets. Further steps of this research work will deepen and clarify this introductory concept (Figure 4).

i. *User needs* are the quality drivers by excellence. The users' judgment affects directly the perception of quality of a given product. User's requirements, preferences and experiences drive the level of ***satisfaction*** reached by an application.

ii. The *mobile environment*, composed by the target devices, mobile networks and mobile operators also pose several requirements on mobile applications. We introduced the evolving and inherent restrictions of mobile targets, which drive the ***suitability*** of a software product to be executed in this environment.

iii. Finally, the *market regulations* are an important source of quality requirements for mobile applications. An application must be in ***compliance*** with the publishing guidelines of the app store so that it may be commercialized through such storefront.

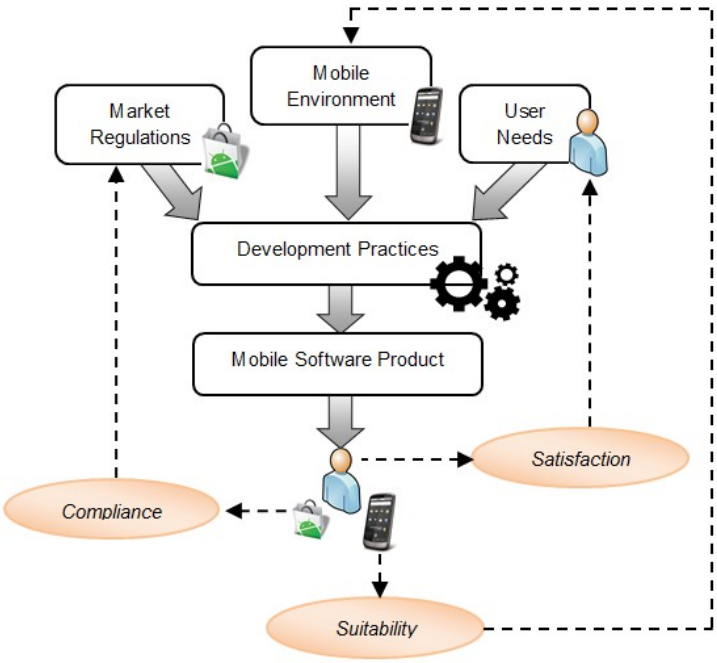


Figure 4. High-level quality drivers of mobile software applications

2.1 Research statement

The objective of this work is to develop the capability of associating the quality requirements and success factors of a mobile product with quality attributes that can be measured and controlled (Figure 5).

Our work hypothesis states:

H.1: The mobile software product can be analyzed to relate measurable parameters of the software product with user expectations and market compliance criteria, delivering an accurate approximation of the product quality. (Corral 2012a)

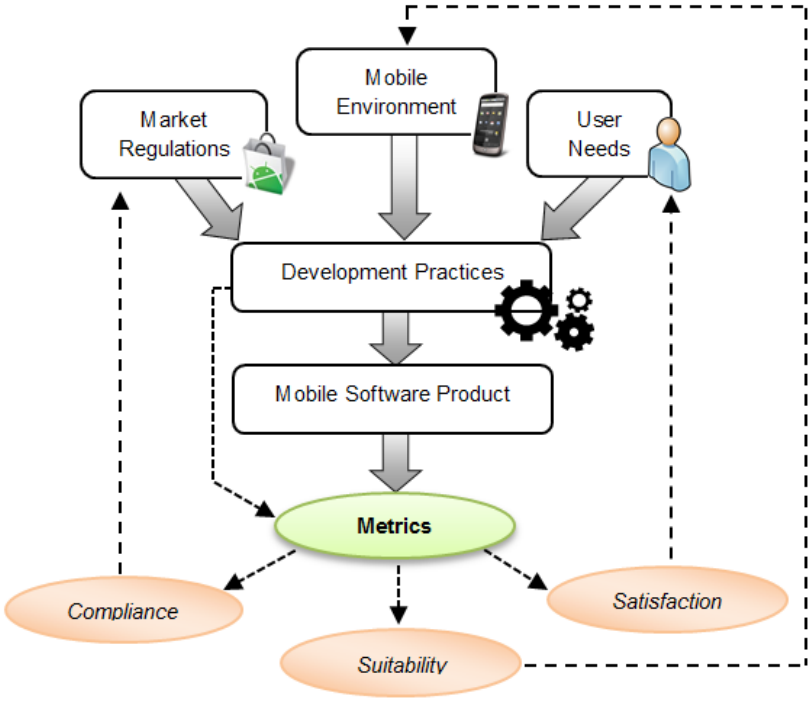


Figure 5. Multi-dimensional approach to measure the quality of mobile software applications

To measure the quality of a mobile application, we have to determine the quality characteristics that describe the software product consistently and uniformly; then, each characteristic may relate to one or more software metrics to assess the software product quantitatively.

We need to look for sources that permit us to approach the concept of quality characteristics. To have a comprehensive understanding of this term, some mature and well-known software product quality standards have been examined: McCall (1977), ISO/IEC 9126 (ISO 2001), ISO/IEC 12207 (ISO 2004), and ISO/IEC 25010 (ISO 2011), etc. Naturally, their effectiveness and applicability on the mobile domain can be challenged as they were created based on "generic" software products and do not necessarily take into account conditions of the mobile software product that are not present or are not relevant for traditional desktop software. Then, it is necessary to modify these quality frameworks to suit the needs of the mobile execution environment.

To do it, we can implement a customization strategy. To illustrate it, we may take the example of the Open Source Maturity Models. These development and appraisal frameworks are specific to the context of Open Source Software (e.g., QualiPSo OMM (Petrinja 2009), MOST (Del Bianco 2009) or QSOS (Petrinja 2010)), but were not created from scratch. Instead, each one picks up from other well-grounded software quality models like CMMi, adapting them to provide material and procedures specific to evaluate Open Source Software. The strategy followed by these models is establishing the boundaries of the OSS scope, pointing out and emphasizing domain-specific needs, and tailoring the quality model to enhance it in terms of applicability and effectiveness. The outcome is a standard-inspired, OSS specific quality methodology.

A similar approach can be undertaken, leveraging a mature product quality strategy and utilizing it as a baseline to incorporate the necessary elements to suit the needs of the mobile software product. In this way, we may select a robust software quality standard and customize it, tailoring its elements to produce a standard-based, mobile-specific quality model. The resulting quality model shall be highly applicable to the mobile domain, and it shall be very effective to single out and analyze the most relevant quality characteristics of the mobile software product.

2.2 Research questions

Currently we do not count on a quality model that considers explicitly the conditions existing in the mobile execution ecosystem. Given the growing relevance of the mobile software applications it is of the highest importance to be able to analyze them to provide a substantiated judgment of their quality characteristics. A disciplined, quantitative notion of the quality of the mobile product may be useful for the development, acquisition, recommendation and deployment of software applications of this kind.

We need to learn from mature quality frameworks and complement them by incorporating domain-specific requirements (e.g., usability matters, market awareness, etc.), and target-specific restrictions (e.g., input methods, screen size, power, etc.). We

should consider that, to appraise the quality of the mobile software product, it is compulsory to have a clear understanding of the attributes that drive it, including having solid foundations about the expectations of all the parts involved.

With the purpose of finding these solutions, we formulate a research question:

RQ: How can mobile software products be analyzed to provide a solid and objective notion of their quality? That is, we set as the main objective of this work the capability of associating the high-level requirements and success factors of a mobile product with quality characteristics that can be measured and controlled.

Due to the complexity of the question, we consider essential to analyze it from different perspectives, so we formulate two research sub-questions:

- *RQ.1: What are the most relevant quality requirements set upon a mobile application?* That is, to survey what is typically expected from a mobile application from the viewpoint of the involved stakeholders (users, execution environment and application markets). Many of the expectations of traditional software hold in the mobile scope; however, it is necessary to deepen this analysis into the mobile domain.
- *RQ.2: How can we measure the quality of a mobile software product?* The answer to this research question will help to refine the answers of RQ.1 by defining a collection of characteristics that aid to approximate the quality of a mobile product considering the quality requirements that are more relevant for the stakeholders.

2.3 Research strategy

Our technical approach is organized in three stages that focus on providing answers for each question established:

i. Research question RQ.1 required to execute comprehensive survey to select the most important quality requirements set upon apps. We conducted a Literature Review of the available development processes and publishing guidelines to

draw an outline of what a mobile application should look for, what constraints it should overcome, how it should perform, and how the product quality should be measured and evaluated. Moreover, we executed a comprehensive survey of mobile application stores, which already count on development and publishing policies that can be considered as a quality reference for the mobile market.

ii. To answer RQ.2, we carried out an analysis that allowed us to relate the mobile-specific quality requirements with the corresponding quality characteristics. The mobile specific quality requirements were previously isolated by RQ.1, based on the needs and considerations of the mobile environment and the mobile market. To characterize such requirements, we implemented suitable quality tools that permitted us to map the software quality requirements with quality characteristics from an appropriate software quality standard that facilitates the organization and prioritization of the quality requirements (Corral 2012b).

iii. Finally, our research strategy considered the implementation of case studies that permitted us to to implement in actual mobile software setting the quality characteristics previously identified, and to assess the quality of such applications based on custom software metrics.

2.4 Standard-based definition of quality characteristics

The implementation of the mobile specific quality strategy relies on the identification of quality characteristics that may be hard to determine. To overcome this issue, we foresee the selection of a mature software assurance framework that can be tailored to the needs given by the mobile ecosystem and the mobile product.

We decided therefore to select the ISO/IEC 25010:2011 quality standard and analyze its quality characteristics and attributes, focusing on those that are applicable to the quality of the mobile software product context. ISO/IEC 25010:2011 defines quality characteristics grouped in two orthogonal dimensions:

i. a "product quality model" that focuses on static properties of software and dynamic properties of the computer system, and

ii. a "quality in use model" that covers the outcome of the interaction when a product is used in a particular context of use.

These double approach to software quality is very suitable to our approach, that considers quality drivers from the development and user viewpoints. Later, specific metrics can be defined utilizing the GQM methodology (Basili 1994) (Figure 6).

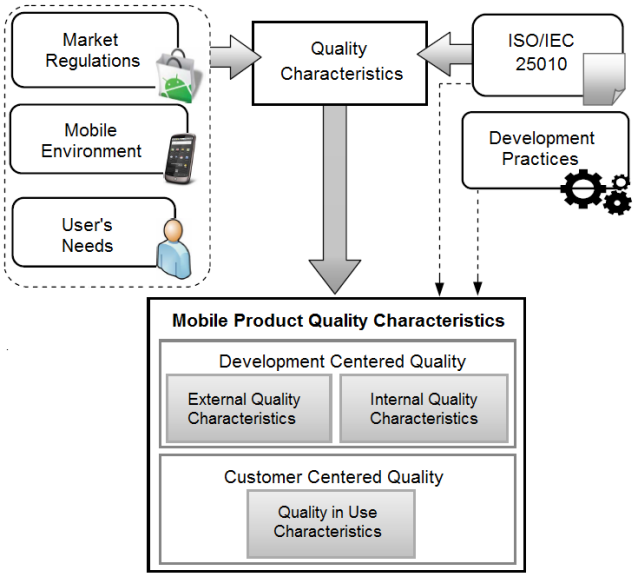


Figure 6. Approach for approximating the quality of the mobile software product

2.5 Research forecast

With this strategy and the related experimentation, we aim to contribute to State of the art on mobile Software Engineering by enabling mechanisms to assure and monitor the quality of the mobile software product, based on real-world quality expectations and real-world market awareness, delivering instruments to identify the characteristics and attributes that enable a well-grounded quantitative analysis. We also aim to provide the means to assist decision making in establishing strategies or recommending practices to optimize the development processes in mobile software projects.

In the long-term, establishing a strategy for software quality assessment for mobile applications will be helpful for researchers, developers and users to:

i. identify quality requirements addressable when designing and implementing the mobile software product, and implement metrics that allow to measure quality attributes and track quality characteristics;

ii. determine the capacity of the mobile software application to meet the requirements and demands posed upon it;

iii. relate the development practices to the level of accomplishment of a given mobile application, using quality metrics.

Our approach relies on applying the ISO/IEC 25010 quality standard to set a software quality reference of for mobile applications, considering in advance the conditions that rise in the mobile environment and that impact the quality of the resulting product. The need of assuring the development of high-quality mobile software products becomes an imperative and demands extensive research and experimentation.

Upon the completion of this research work, we will be able to identify the most relevant quality expectations of mobile software products, and analyze them based on standard quality characteristics that can be evaluated quantitatively through measurable quality attributes.

2.6 Chapter bibliography

(Basili 1994) Basili, V.; Caldiera, G.; Rombach D.; (1994). The Goal Question Metric approach. In *Encyclopedia of software engineering* pp. 528-532. Wiley. doi:10.1002/0471028959.sof142

(Corral 2012a) Corral, L.; (2012) Using software quality standards to assure the quality of the mobile software product. In *Proceedings of the 3rd annual conference on systems, programming, and applications: software for humanity (SPLASH '12).* pp. 37-40. ACM. doi:10.1145/2384716.2384734

(Corral 2012b) Corral, L.; (2012) Standard-based strategy to assure the quality of the mobile software product. In *Proceedings of the 3rd annual conference on systems, programming, and applications: software for humanity (SPLASH '12).* pp. 95-96. ACM. doi:10.1145/2384716.2384755

(Del Bianco 2009) Del Bianco, V.; Lavazza, L.; Morasca, S.; & Taibi, D.; (2009) Quality of open source software: The QualiPSo trustworthiness model. In *Proceedings of the 5th IFIP WG 2.13 International Conference on Open Source Systems (OSS 2009)*. IFIP AICT, vol. 299. pp. 199-212. Springer Berlin / Heidelberg. doi:10.1007/978-3-642-02032-2_18

(ISO 2001) International Organization for Standardization (2001) *ISO/IEC 9126:2001. Software engineering - Product quality.* Available online: http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=22749 (Accessed on September 7th, 2013).

(ISO 2004) International Organization for Standardization (2004) I*SO/IEC/IEEE 12207. Systems and software engineering - Software life cycle processes.* Available online: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4475826 (Accessed on September 7th, 2013).

(ISO 2011) International Organization for Standardization (2011) *ISO/IEC 25010:2011. Systems and software engineering. Quality requirements and evaluation system and software quality models.* Available online: http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=35733 (Accessed on September 7th, 2013).

(McCall 1977) McCall, J. A.; Richards, P. K.; Walters, G. F.; (1977). *Factors in software quality.* Technical Report. National Technical Information Service.

(Petrinja 2009) Petrinja, E.; Nambakam, R.; & Sillitti, A.; (2009) Introducing the Open Source Maturity Model. In *Proceedings of the 2009 ICSE Workshop on Emerging Trends in Free/Libre/Open Source Software Research and Development (FLOSS '09).* pp. 37-41. IEEE Computer Society. doi:10.1109/FLOSS.2009.5071358

(Petrinja 2010) Petrinja, E.; Sillitti, A.; & Succi, G.; (2010) Comparing OpenBRR, QSOS and OMM assessment models. In *Proceedings of the 6th IFIP WG 2.13 International Conference on Open Source Systems (OSS 2010).* IFIP AICT, vol. 319. pp. 224-238. Springer Berlin / Heidelberg. doi:10.1007/978-3-642-13244-5_18

Chapter 3: Software Assurance for Mobile Products: State of the Art

We discussed that the mobile ecosystem is complex, competitive and fault prone. This situation poses to researchers and practitioners the question of how to develop software products able to succeed in this domain. Several works discuss different approaches to execute development, measurement and assurance practices for mobile software projects. We should analyze them to find what methods have been introduced, accompanied if possible by successful case studies of their utilization.

3.1 State of the Art and the Practice: Research questions

We want to know the current status of the practices that concern on guaranteeing the quality of the mobile software product based on software quality assurance practices; for instance, software life cycles, development methodologies, and others. To this end, we formulated as research sub-questions:

RQ1.1: What quality assurance practices have been proposed to address the specific needs of the mobile software product? The answer to this research question provides us with a notion of the State of the Art, supplying a collection of research papers that have proposed different approaches to address the quality needs of the mobile software product, spanning through the different stages of the software development life cycle. In

addition, we are interested on analyzing the applicability of these practices in a real setting. This gives place to our second research question:

RQ1.2: What is the evidence of use of the mobile-specific software quality practices in a productive setting? The answer to this research question provides us with a notion of the State of the Practice, supplying the evidence that endorses the utilization and effectiveness of the proposed practices in a productive (i.e., industrial, business, practitioner) environment.

To provide a sustained answer to the research questions, our approach was twofold: First, we executed a survey on the major repositories of Computer Science literature to identify the research works that propose assurance practices for mobile software. Second, we looked for instances that cite these practices to have a better understanding on how these practices can be implemented in a real case.

3.2 Literature review strategy

We surveyed three major Computer Science digital libraries (ACM Digital Library, IEEE Xplore and ScienceDirect) looking for research articles that introduce methodological, practical and quantitative assurance practices applicable to a generic mobile software project. The period chosen was from 2002 to 2013, in accordance with the ramp up, development and consolidation of smart phones as target platform for mobile software. We put particular attention on the period from 2008 to 2013, which comprises the time span of the release of iOS and Android OS, the most influential mobile operating systems.

The selected keywords aimed to isolate research works concerning the development and quality management in a mobile software project (e.g., Mobile Software Engineering, Mobile Software Development Process, Mobile Software Quality, Mobile Software Assurance, and Mobile Software Metrics). To make the selection of relevant works, we established as inclusion criteria:

i. Research works should present practices specific and applied for mobile software;

ii. Research works should focus on assuring the conduction of a generic mobile software project;

iii. Research works should have been put in practice preferably in at least one case study;

iv. Research works should have been included in an international journal or conference proceedings.

To collect the evidence of use of the practices in a productive environment, we looked for additional instances (research papers, surveys, web sites, technical and experience reports) that cite the selected papers.

3.3 Review of software practices for mobile software

After surveying the digital libraries, 23 highly relevant research papers were selected. From this body of research, we could distinguish three layers of practices: process oriented, product oriented, and implementation oriented. Each family of works concerns on different levels of the software development:

- *Process oriented practices:* they pursue to furnish step-by-step processes prescribing the necessary activities for the diverse stages of the development life cycle, from the requirements elicitation through the final delivery. This family of works is described in section 3.3.1.

- *Product oriented practices:* they focus on introducing techniques to analyze and measure the mobile software product from an artifact-oriented perspective. This family of works is described in section 3.3.2.

- *Implementation oriented practices:* they concentrate on proposing architectural guidelines, design patterns and development best practices to implement on a mobile software project. This family of works is described in section 3.3.3.

To answer RQ1.1, our strategy to understand the State of the Art focuses on providing an individual analysis of each of the families of works. A summary of the selected works is shown in Table 1.

Table 1. Summary of research works presenting mobile software development practices

Family or Practice	Sub-Family or Practice	Number of Works
Process oriented	Agile	5
	Non-Agile	3
Product oriented	Standard-based	2
	Metric-based	3
	Test-based	3
Practice Oriented	Design Patterns	5
	Best Practices	2

3.3.1 Process oriented practices

The fast-paced mobile market sets the need of having lightweight processes that facilitate the change and the adoption of emerging trends. An effective development strategy should be strong enough to consider the quality drivers of the mobile ecosystem and to cope with the markets competitiveness, but at the same time it should be enabling technologies. Our review identified some process-oriented research works, which can be classified in Agile and Non-Agile approaches.

a) Agile Approaches

In 2003, it was discussed for the first time the suitability of Agile practices to fulfill the objectives of the mobile software development (Abrahamsson 2003). Later, it was shown a thorough mapping between the Agile home ground themes with several development traits observed in mobile software. This mapping permitted to outline why Agile might

be a competent solution for implementing development processes in this domain, based on characteristics like collaboration in small teams, reduction of development times, management of ongoing changes of requirements, dealing with the variety of target platforms, and the assumption a small-sized, non-critical end product (Abrahamsson 2005). According to our selection criteria, we found in literature some methodologies that undertake the Agile approach to design a comprehensive mobile-specific software development process:

- *Mobile-D:* It was the first attempt to incorporate Agile in the development of mobile applications. It was introduced by Abrahamsson et al. (2004) as a development methodology inspired by Extreme Programming, Crystal Methodologies and the Rational Unified Process. It is recommended to be used by a small, co-located team, working in a short development cycle. It is structured in five phases (Explore, Initialize, Productionize, Stabilize and System Test/Fix), sequentially arranged following a generic software development process. Despite of its sequential organization, Mobile-D encourages iterations, after which a functional product is released.

- *MASAM:* Mobile Application Software development based on Agile Methodology was proposed by Jeong et al. (2008) and it is based on Extreme Programming, Agile Unified Process, the Rational Unified Process and the Software and Systems Process Engineering Meta-model. MASAM follows a software life cycle based on the Agile approach, proposing a simple life cycle formed by four phases (Preparation, Embodiment, Developing and Commercialization). The structure and detailed implementation of MASAM show a strong tie with Mobile-D and only introduces minor variations, making its original contribution rather marginal.

- *Hybrid:* Rahimian and Ramsin (2008) promoted a conjunction of Agile and plan-based methodologies. To structure their methodology, it is proposed to baseline a generic software development life cycle and to customize it with a merge of Agile practices and principles of New Product Development. The outcome is a Hybrid Methodology Design Process. It defines an "Iterative Design Engine", that

is, a process that designs, models, integrates and reviews all software components, and finishes with market testing to guarantee fitness for commercialization.

- *Scrum:* Schaff and Verma (2010) covered the use of Scrum for the development of mobile applications. Scrum is an iterative and incremental frame-work commonly used in combination with other Agile practices. It uses iterations of fixed duration (typically one to four weeks) called sprints. At the beginning of each sprint, the development team commits to complete a certain number of tasks selected from the Product Backlog and documents them in a Sprint Backlog. After this, the Scrum team decides how much work they will commit to complete in the next sprint, until the Product Backlog is finished and the product is delivered.

- *Scrum Lean Six Sigma (SLeSS):* It is an approach of integrating Scrum and Lean Six Sigma, proposed by Cunha et al. (2011). This methodology enables the achievement of performance and quality goals, improving the processes in a statistically controlled basis. SLeSS picks up from the Scrum methodology, but it pursues a combination of the effort and consistent deliveries of the sprints with the continuous process analysis and improvement model represented by the 5-phase DMAIC methodology (Define, Measure, Analyze, Improve and Control). The implementation of SLeSS considers an incremental approach, in which the Agile philosophy (Scrum) is adapted to coexist with the planned-based methodology (Lean Six Sigma).

b) Non-Agile Approaches

The Agile approach was first introduced under the premise that mobile applications are generally simple, dedicated to accomplish a very restricted number of actions, usually non-critical, and can be developed by small teams in short periods. Nevertheless, the evolution of the mobile software product and its execution targets have left behind some of these assumptions. More recently, some non-Agile development and

assurance approaches were introduced. According to our selection criteria, we found in literature the following non-Agile methodologies:

- *M-Business Procedure Model:* The m-business procedure model was introduced by Glissman et al. (2005) for the development of mobile user interfaces. The model is based on well-established software engineering and human computer interaction design principles. It is concreted with detailed development recommendations regarding mobile-specific issues, concentrated in five sequential stages: Needs, Requirements, Conceptual design, Physical design and Implementation.

- *Mobile Development Process Spiral:* The Mobile Development Process Spiral was proposed by Nossier et al. (2012) to utilize a custom model to integrate mobile-specific usability matters into existing application development processes. This process model baselines the Boehm Spiral Model and adapts it to the context of mobile software. The process spiral works as a user- centered iterative process model structured on several iterations where the requirements are addressed by Independent design, Development, Testing and Planning phases. The authors propose this methodology for large, expensive, and complicated projects as an aid for risk reduction.

- *Intel Mobile Application Development Framework:* Even though it was not found in the referred libraries, we considered interesting to include the Intel Mobile Application Development Framework (Doolittle 2012). It is an enterprise-oriented effort that focuses on evaluating the suitability of mobile applications to generate business value within a company. It prescribes guidance documentation, enabling technologies and supporting resources for conducting projects that adhere better to the standards and best practices of the organization.

3.3.2 Product oriented practices

A product-oriented family of works concentrates on proposing practices to analyze, measure and test mobile applications. These practices consider upfront the characteristics of the mobile execution environment, and promote the customization of general-purpose product quality practices to suit the needs of the mobile ecosystem.

- *Standard-based software product assessment:* Spriestersbach and Springer (2004) summarized the typical challenges in the development of mobile web applications, and related such challenges into the quality characteristics described the ISO/IEC 9126 standard. Mantoro (2009) utilized as well the ISO/IEC 9126 standard to evaluate several attributes on and context-aware applications (e.g., usability matters, network connection, target devices, etc.) that also hold for general-purpose mobile applications. Both associations may lead to potential adjustments in the ISO/IEC 9126 model, focusing on relevant quality attributes of mobile software applications.

- *Metric-oriented product assessment:* Ryan and Rossi (2005) stated that few software metrics consider the unique characteristics of mobile applications, unveiling an important niche for future research on mobile software engineering. In this work, the authors propose a set of metrics to monitor statically source code attributes related to the efficiency of mobile applications. Hussain and Ferneley (2008) used the GQM approach to produce a metric for the evaluation of usability on mobile applications. Pandi and Charaf (2013) introduced performance metrics as an input for resource management. In this study, they presented an architecture for performance measurement, accompanied by key performance metrics.

- *Test-based product assessment:* Dantas et al. (2009) published a review of testing requirements specific to applications developed for mobile devices. Some of the proposed requirements are: testing of mobile applications in both emulators and mobile devices; ensuring that mobile applications must not harm

anything already deployed on the device; testing mobile applications according to the mobile context limitations. Liu et al. (2010) proposed an adaptive random test case generation technique to produce black-box test cases for mobile applications. Their results show that such technique can both reduce the number of test cases and the time required to complete the tests. Amalfitano et al. (2013) discussed on the consideration of context events for testing mobile apps, since this kind of applications are commonly event-driven systems that should take into account both context and GUI events. Their approach is based on the definition of reusable event patterns for the automatic generation of test cases.

3.3.3 Implementation oriented practices

The last group of research works report design practices or implementation guidelines for mobile software. They do not describe a comprehensive software development framework, but they provide hands-on development guidelines, design methodologies and best practices that can be applied following any development process.

- *Design methodologies:* From the design standpoint, we found a high number of research papers that focus on recommendations to design user interfaces that suit better the mobile display, input and output means. We did not consider this family of works, since they were in the field of human-computer interaction rather than in software engineering. Instead, we concentrated on works that provide architecture patterns (Ihme 2005), (Kamthan 2008) to facilitate the implementation of mobile applications. Also, other group of works presents design patterns (La 2011), (Kim 2011), (Sokolova 2013) for mobile applications, based mostly on the MVC (model-view-controller) approach.

- *Best Practices:* A group of research papers attempts to generate best practices for the implementation of mobile software products. De Sa and Carrico (2008) elaborated a report of recommendations to address the challenges that emerged through the data gathering, prototyping and evaluation of mobile

applications. Marinho and Resende (2012) produced a relationship between several general-purpose quality models (McCall, ISO, etc.), with the specific quality factors and the best practices that are more appropriate to implement a mobile application. Other corporate mobile development best practices include those from the W3 Consortium (W3C 2010) and from the Unified Test Initiative (UTI 2011).

3.4 Literature review results

To provide an answer to RQ1.2, we extended the analysis to the discussion of the accomplishment and evidence of use of the reviewed works. Originally, our survey identified research works that provide different practices that aim to assure the production of a fully functional mobile software application. When available, the research contributions make reference to case studies that illustrate the fitness of those methods to assist the development and delivery of a real product.

However, it is not realistic to claim that these practices are applicable and effective in a real environment without supplying additional evidence. Furthermore, there have been significant changes in the conditions of the mobile ecosystem during the time frame in which the mobile assurance practices were released. For instance, at the beginning of the time span selected for our review, concepts like location-based services, app stores, iOS, Android and other current key terms were emerging concepts or did not exist at all. As a strategy to solve our second research question, we discussed the matter in three thematic areas, suitability, evidence and evolution (Corral 2013a).

3.4.1 Suitability

The reviewed methodologies claim effectiveness on addressing the needs and constraints of the mobile software ecosystem. We noted, though, that for the process-oriented research works it is hard to argue for a direct effect on the end product at the level of abstraction they are presented. Considering the constraints of the mobile environment, it is not clear what problems can be injected if a general-purpose

software development process is used, or why a software development process would help to resolve these issues. For instance, if we consider a scenario in which one has to design an application to work in an environment with poor and intermittent connectivity, what development process (Mobile-D, Scrum, Spiral etc.) should be used? Why this choice would be an important factor in the success of the final outcome? Such questions are generally left open at this level.

With respect to product measurement, with the introduction evaluation metrics and other quantitative indicators, the body of knowledge on software quality for mobile applications provides the developer with instruments to assess, from a practical point of view, the success and fitness of use of the end product from an artifact-driven viewpoint that explicitly takes into account the conditions and limitations of the mobile ecosystem. Finally, the implementation-oriented level gives preference to practices that take into account, from a practitioner's approach, the environment conditions that affect the mobile product, including interface design, usability guidelines, design patterns, and others. These practices concentrate effectively on the end product and can be exercised regardless of the software development process that is being followed.

We can suggest that only the conjunction of available assurance practices at the three levels collaborates to furnish a robust development process that is appropriate to meet the business needs and environment constraints of the mobile ecosystem. Product oriented practices stand in the middle to monitor the activities and supply the data that support the measurement tasks required by the high level development process. At a hands-on, level, the architecture recommendations, design guidelines, and best practices provide implementable practices that aid to create a suitable product (Figure 7).

In a different front, we should remark the high influence of application markets in the development practices and quality attributes of the mobile software product. Mobile application stores (e.g., Google Play, iOS App Store, etc.) pose requirements or publishing policies that a product must meet to be included in the catalog listings.

Under this rationale, the publishing guidelines prescribed by app stores represent the standard

quality level for a mobile app to enter into service. Publishing policies are fundamental quality criteria for mobile developers, since the lack of compliance may cause rejection or exclusion from the app store. In spite of this, we noted that publishing guidelines from app stores are typically overlooked (i.e., not explicitly mentioned) by the three layers of practices.

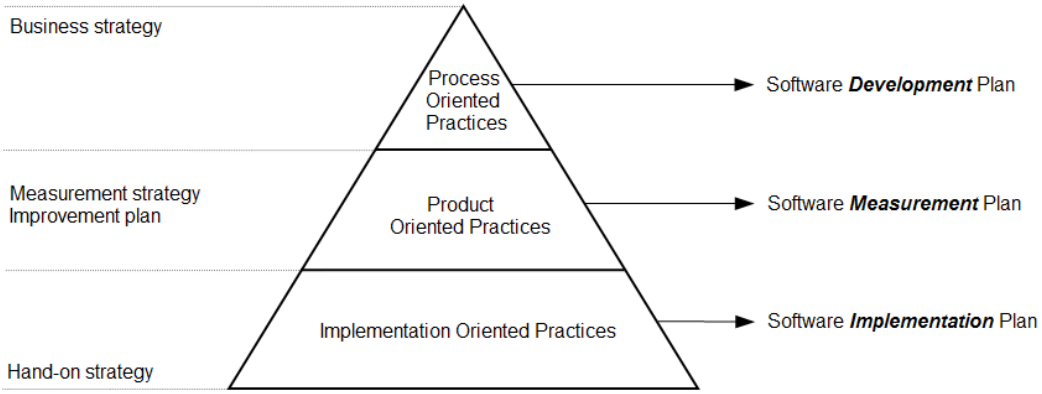


Figure 7: Processes, product assurance practices and implementation practices

3.4.2 Evidence

One of the goals of this survey is to describe whether there is enough evidence about the real application of the proposed practices in a productive setting. The research contributions that present assurance practices usually include an academic or industrial experiment to validate their approach. Still, to claim the effectiveness and success of each practice it is necessary to supply empirical evidence of their actual spread and implementations. We surveyed the selected scientific libraries and the World Wide Web looking for research papers, technical reports or documentation that explicitly reports out the utilization of each of our selected practices.

- *Process Oriented Practices:* Besides Mobile-D, the rest of the process oriented frameworks cannot argue to have a major adoption in a real production environment. Mobile-D keeps records about software projects developed using

this methodology, including works carried out both in research and large industry settings (ENERGI, F-Secure, Nokia, Philips) (Pikkarainen 2005), (Hedberg 2006), (Abrahamsson 2007). Scrum, SLeSS, m-Business and Mobile Spiral present only one case study as part of the validation of their work, but references to them do not show further implementations. MASAM, Hybrid Methodology and the Intel Framework do not show a case study, and are not cited by any other report. This situation challenges seriously their feasibility and potential success in a real setting.

- *Product Oriented Practices:* The standard based approach introduced by Spriestersbach (2004) and Mantoro (2009) enjoys of high popularity in a number of research works that base their quality strategy in ISO/IEC 9126 and more recently, in ISO/IEC 25010. However, the explicit references to these research works are limited. The metric oriented approach introduced by Ryan and Rossi (2005) is also seminal in further efforts that have as well proposed new metrics that at times specialize in a certain sub-area, like usability, performance, etc. Finally, the test-based approach permitted to identify key attributes of the mobile software product that is recommended to verify upfront. In summary, even though the product-oriented family of works has a high number of citing works; these papers make reference to the need of having product specific assurance practices for mobile applications, but do not tend to show a direct implementation of the proposed ones.

- *Implementation Oriented Practices:* The design practices reported by Ihme (2005), originated in the same research group that created Mobile-D (Abrahamsson 2004) are the most spread in scientific literature. The rest of the analyzed papers have a very limited number of citations that do not permit to evaluate their real implementation in other settings. The best practices of De Sa (2008) are also highly cited, but the number of works that make direct reference of their utilization is low.

Table 2. Mobile software assurance practices and their implementations

Family of Practice	Practice	Year	Cited by	Case Studies
Process, Agile	Abrahamsson et al	2004	71	16
	Jeong, et al.	2008	11	0
	Rahimian and Ramsin	2008	22	0
	Scharff and Verma	2010	16	3
	Cunha et al.	2011	6	1
Process, Non-Agile	Glissman et al.	2005	18	1
	Nossier et al.	2012	3	1
	Intel	2012	2	0
Product, Standard	Spriestersbach and Springer	2004	16	3
	Mantoro	2009	5	0
Product, Metrics	Ryan and Rossi	2005	27	3
	Hussain and Ferneley	2008	12	1
	Pandi and Charaf	2013	0	1
Product, Test	Dantas et al.	2009	11	1
	Liu et al.	2010	14	1
	Amaltano et al.	2013	0	0
Implementation, Design Patterns	Ihme and Abrahamsson	2005	18	5
	Kamthan	2008	0	0
	La et al.	2011	4	0
	Kim and Park	2011	2	1
	Sokolova et al.	2013	0	1
Implementation, Best Practices	De Sa and Carrico	2008	39	4
	Marinho and Resende	2012	1	0

Table 2 shows a summary of the documented instances of implementation of each methodology, including the case studies presented as part of the research paper in which they are introduced. Citation count was taken from the digital library when available; otherwise it was reviewed in Google Scholar. We count a case of utilization if

the referencing paper explicitly mentions that they utilize the methodology or technique introduced by the referenced research work.

We expanded this analysis with a review of additional field studies, that is, development surveys, attempting to identify substantiation of the usage of a methodology, pattern or other development practices in true mobile software projects (Rajapakse 2008), (Avram 2011), (Wangenheim 2011). Unfortunately, the reviewed studies focus their interest on analyzing the operating system of choice, software development kits, type of applications produced, multi-platform deployment and other topics. Although they suggest a clear trend on shortening the development cycle and broaden the impact of the end product, further work is required to unveil the utilization of a consistent development methodology or assurance practice (Marko 2012).

3.4.3 Evolution

When the first mobile-specific development practices were introduced, they considered the Agile approach as the best fit for mobile development. Nevertheless, in those days the mobile business and development and execution environment were different from the current one. Table 3 shows the mapping between the Agile home ground themes and the characteristics of the mobile software, made available by Abrahamsson (2005).

A decade of evolution in the mobile domain (software, hardware and business models) has brought significant advancements; therefore the current applicability of this mapping is controversial and invites to conduct an up-to-date discussion. To name only some of the differences of the current status of the mobile domain, we identify: While hundreds of new mobile models are still released each year, mobile developers also have well settled operating systems that have development kits (SDK) and APIs that facilitate the interaction with new device models.

Table 3. Mapping of Agile ground themes and traits on mobile software development

Ideal Agile Characteristic	Mobile Software Development
High environment volatility	Dynamic environment: hundreds of new mobile phones published each year.
Small development teams	Majority of mobile software is developed in micro or SME companies or development teams.
Identifiable customer	Potentially unlimited number of end-users.
Object-oriented development	Java and C++ are mainly used.
Non-safety critical Software	Majority of existing mobile software is for entertainment purposes. Mobile terminals are not reliable.
Application level software	Mobile applications are stand-alone applications.
Small systems	Size of mobile applications varies, but generally they are less than 10,000 lines of code.
Short development cycles	Generally mobile applications and services can be developed within 1-6 month time frame.

- Mobile software is still developed by small teams and small-medium enterprises, but currently it is also part of major developments that involve large corporate teams.

- Nowadays, mobile applications include not only stand-alone applications but also interaction with other systems, collaboration tools, using heavily network and hardware resources, etc. This also implies that the mobile software product is not anymore small by definition.

- The range of applications deployed on cellular telephones now includes healthcare monitors[4], mobile banking[5] or earthquake alerts[6] that are required to meet strict standards to enter into service and cannot be categorized as non-critical software.

[4] http://www.fda.gov/NewsEvents/Newsroom/PressAnnouncements/ucm263340.htm

[5] http://www.sans.org/reading_room/whitepapers/ecommerce/security-mobile-banking-payments_34062

[6] http://www.economist.com/blogs/americasview/2012/04/earthquake-warnings-mexico-city

We observe that some Agile-based mobile software development frameworks try to enhance their methods by adapting practices from plan-based methodologies, for example, project documentation, traceability records, and other considerations. For instance, the latest Agile proposal, SLeSS, is an approach that tries to relieve several shortcomings from Agile by applying statistically-based quality control. This represents a complex merge of two different viewpoints: light-weight development practices (Scrum) and heavy quality control methodologies (Six Sigma). Following this idea, other recent development methodologies like the Mobile Spiral or the Intel Model have completely relegated the Agile approach.

These examples reflect an identified decline in considering Agile as a silver bullet (Janes 2013), (Corral 2013b) and instead promote a savvy strategy to decide when to use which practice evaluating its advantages and disadvantages. Likewise, our research in mobile computing permitted us identify examples of mission critical applications, in which a mobile device becomes a key actor in safety and security scenarios (Corral 2011a), (Corral 2011b) that require mobile-enabled software to follow a plan based strict development methodology.

The current status of the mobile-specific assurance practices provides software engineers with a solid body of knowledge to combine the three layers of practices to decide what approach to apply: Agile practices, plan-based methodologies, stringent product-oriented quality inspections, software metrics, design patterns or implementation techniques.

3.5 Open items

Our literature survey provided a clear idea about the available quality assurance practices for mobile software, their scope and taxonomy. Nonetheless, the analysis of the utilization of the reviewed assurance practices showed that their implementation in a real setting is still limited. The academic citation count provides only a general view of finalized projects that exercised one or more of the published practices, but to have a more robust answer on the matter there is a clear need of conducting evidence-based

research that unveils what mobile development practices are actually being used. These insights may be gained by means of industrial surveys, interviews with mobile software managers and other empirical studies, and from the proactive discussion with the scientific community.

Development guidelines from major application stores have been ignored by scientific literature. Mobile quality practices should consider upfront the requirements from the user, the execution ecosystem, and the application market. In consequence, ignoring the market quality drivers is a major shortcoming for the current body of knowledge. To address this need, we recommend an analytical survey of the development policies of the major application stores to extract the most relevant quality assurance requirements. Then, we may be able to recommend process and product oriented quality assurance tasks to guarantee the fulfillment of the quality attributes that are most relevant for real mobile app stores. With this research field open, further efforts should concentrate in setting improved mechanisms to assure the quality of the mobile software product from a market-aware point of view.

Finally, the works of Hammershoj (2010), Wasserman (2010), Gasimov (2010) Dehlinger and Dixon (2011), and Muccini et al. (2011) provide important research directions on mobile Software Engineering, identifying major concerns that may be covered by mobile development processes in the near future: user experience, integration with cloud computing, mission critical development, energy aware implementations, multi-platform development and mobile specific testing practices are some of the recommended topics for further research. In summary, from the point of view of the analyzed development practices, we identify:

a) Platform-specific software quality assurance

Defects in mobile applications should not escape detection. Mobile quality assurance practices can be improved in a number of ways, especially with regard to target-specific needs. The analyzed methods show a diligent effort on general-purpose process assurance, but they still lack considerations of the target environment at hand.

Currently, a limited number of works on mobile-specific quality practices can be found, thus we can consider this area as a rich field for further research.

b) Platform-specific software product measurement

Mobile software development practices should consider as well environment-specific circumstances to enrich their quality activities in a quantitative approach. However, in the field of software measurement for mobile applications (e.g., mobile software metrics) there is not much discussion published up to now. Thus, to address the two identified major needs, we envision customizing a strong software quality framework with mobile-specific needs to measure the quality of the mobile software product and determine its capacity to meet defined requirements. Based upon a mature quality standard, we may be able to recommend process and product oriented quality assurance tasks, design and implement metrics to measure quality attributes, identify acceptance, release and acquisition criteria (e.g., recommendation for inclusion in an application market), etc. With these research fields open, further sections of this thesis concentrate in setting improved mechanisms to assure the quality of the mobile software product, including defining metrics for quantitative quality assurance.

3.6 Summary and conclusions

The conducted literature survey permitted us to draw the current picture of the State of the Art and the Practice in the area of quality assurance practices for mobile software.

To answer our research question RQ1.1, regarding the State of the Art, we were able to gather a number of software assurance practices for mobile systems, and to determine that these practices are divided in three layers: software development processes, software product assurance practices, and software implementation practices. The methodologies and techniques reviewed in this paper call attention on the necessity of adapting general-purpose processes and practices to the specific needs of the mobile ecosystem. Current mobile software quality practices have evolved by adapting practices from Agile and plan-based methodologies, incorporating product

measurement, best practices, testing techniques, design patterns, and other similar considerations.

For our research question RQ1.2, regarding the State of the Practice, we investigated the supporting evidence about the utilization and effectiveness of the proposed practices in a productive setting, extracting the citing works and isolating the relevant implementations. The number of research works and reports that make reference to the proposed methodologies is high for a limited number of practices only. For the rest, citation count and the overall referencing is small, which suggests that the practitioner community has followed a different quality assurance paradigm that has not been covered in detail by the software engineering research community.

The evolution of mobile platforms from being a simple communication and entertainment tool toward becoming the primary end-user computing equipment requires researchers and practitioners to understand the context of the mobile domain to create the best strategies to develop and assure mobile software. Mobile Software Engineering still faces an extensive work to deter mine what are the best processes and practices that facilitate the creation of high quality mobile software products.

3.7 Chapter bibliography

(Abrahamsson 2003) Abrahamsson, P.; Warsta, J.; Siponen, M.T.; & Ronkainen, J.; (2003) New directions on agile methods: a comparative analysis. In *Proceedings of the 25th International Conference on Software Engineering 2003 (ICSE'03).* pp. 244-254. IEEE Computer Society. doi:10.1109/ICSE.2003.1201204

(Abrahamsson 2004) Abrahamsson, P.; Hanhineva, A.; Hulkko, H.; Ihme, T.; Jäälinoja, J.; Korkala, M.; Koskela, J.; Kyllönen, P.; & Salo, O.; (2004) Mobile-D: an agile approach for mobile application development. In *Proceedings of the 19th annual ACM SIGPLAN conference on Object-oriented programming systems, languages, and applications (OOPSLA '04).* pp. 174-175. ACM. doi:10.1145/1028664.1028736

(Abrahamsson 2005) Abrahamsson, P; (2005) Mobile software development – the business opportunity of today (Keynote). In *Proceedings of the International Conference on Software Development.* pp. 20-23.

(Abrahamsson 2007) Abrahamsson, P; (2007) Agile software development of mobile information systems. In *Proceedings of the Advanced Information Systems Engineering Conference.* Lecture Notes in Computer Science, vol. 4495. pp. 1-4. Springer Berlin / Heidelberg. doi:10.1007/978-3-540-72988-4-1

(Amalfitano 2013) Amalfitano, D.; Fasolino, A.R.; Tramontana, P.; & Amatucci, N.; (2013) Considering context events in event-based testing of mobile applications. In *Proceedings of the IEEE Sixth International Conference on Software Testing, Verication and Validation Workshops (ICSTW).* pp.126-133 doi:10.1109/ICSTW.2013.22

(Avram 2011) Avram, A.; (2011) *A survey on mobile development.* InfoQ. Available online: http://www.infoq.com/news/2011/05/A-Survey-on-Mobile-Development (Accessed on January 20th, 2013).

(Corral 2011a) Corral, L.; Sillitti, A.; & Succi, G.; (2011) Preparing mobile software development processes to meet mission-critical requirements. In *Proceedings of the 2nd Annual Workshop on Software Engineering for Mobile Application Development. In conjunction with MOBICASE 2011.* pp. 9-11.

(Corral 2011b) Corral, L.; Sillitti, A.; & Succi, G.; (2011) Managing TETRA channel communications in Android. In *Proceedings of the 13th International Conference of Enterprise Information Systems (ICEIS 2011).* vol. 3. pp. 307-312. SciTePress.

(Corral 2013a) Corral, L.; Sillitti, A.; & Succi, G.; (2013) Agile software development processes for mobile systems: Accomplishment, evidence and evolution. In *Proceedings of the 10th International Conference on Mobile Web Information Systems (MobiWIS 2013).* Lecture Notes in Computer Science, vol. 8093. pp. 90-106. Springer-Verlag Berlin / Heidelberg. doi:10.1007/978-3-642-40276-0_8

(Corral 2013b) Corral, L.; Sillitti, A.; & Succi, G.; (2013) Software development processes for mobile systems: Is Agile really taking over the business? In *Proceedings of the 1st International Workshop on Mobile-Enabled Systems (MOBS 2013), in connection with ICSE 2013.* pp. 19-24. IEEE. doi: 10.1109/MOBS.2013.6614218

(Cunha 2011) da Cunha, T.F.V.; Dantas, V.L.L.; & Andrade, R.M.C.; (2011) SLeSS: A Scrum and Lean Six Sigma integration approach for the development of sofware customization for mobile phones. In *Proceedings of the 2011 25th Brazilian Symposium on Software Engineering (SBES).* pp. 283-292. IEEE Computer Society. doi:10.1109/SBES.2011.38

(Dantas 2009) Dantas, V.L.L.; Marinho, F.G.; da Costa, A.L.; & Andrade, R.M.C.; (2009) Testing requirements for mobile applications. In *Proceedings of the 24th International Symposium on Computer and Information Sciences, 2009 (ISCIS 2009)*. pp. 555-560. IEEE. doi:10.1109/ISCIS.2009.5291880

(Dehlinger 2011) Dehlinger, J.; & Dixon, J.; (2011) Mobile application software engineering: Challenges and research directions. In *Proceedings of the 2nd Annual Workshop on Software Engineering for Mobile Application Development. In conjunction with MOBICASE 2011.* pp. 27-30.

(Doolittle 2012) Doolittle, J.; Moohan, A.; Simpson, J.; & Soanes, I.; (2012) *Building a mobile application development framework.* Intel. Available online: http://communities.intel.com/docs/DOC-19555 (Accessed on January 20th, 2013).

(Gasimov 2010) Gasimov, A.; Chuan-Hoo, T.; Chee W.P.; & Sutanto, J.; (2010) Visiting mobile application development: what, how and where. In *Proceedings of the 9th International Conference on Mobile Business and Ninth Global Mobility Round (ICMB-GMR).* pp.74-81. doi: 10.1109/ICMB-GMR.2010.20

(Glissmann 2005) Glissmann, S.; Smolnik, Stefan; Schierholz, R.; Kolbe, L.; & Brenner, W.; (2005) Proposition of an m-business procedure model for the development of mobile user interfaces. In *Proceedings of the International Conference on Mobile Business, 2005 (ICMB 2005).* pp. 308-314. IEEE Computer Society. doi:10.1109/ICMB.2005.83

(Hammershoj 2010) Hammershoj, A.; Sapuppo, A.; & Tadayoni, R.; (2010) Challenges for mobile application development. In P*roceedings of the 2010 14th International Conference on Intelligence in Next Generation Networks (ICIN).* pp.1-8. IEEE. doi:10.1109/ICIN.2010.5640893

(Hedberg 2006) Hedberg, H.; & Iisakka, J.; (2006) Technical reviews in Agile development: Case Mobile-D. In *Proceedings of the Sixth International Conference on Quality Software, 2006 (QSIC 2006).* pp. 347-353. IEEE. doi:10.1109/QSIC.2006.63

(Hussain 2008) Hussain, A.; & Ferneley, E.; (2008) Usability metric for mobile application: a goal question metric (GQM) approach. In *Proceedings of the 10th International Conference on Information Integration and Web-based Applications & Service.* pp. 567-570. ACM. doi:10.1145/1497308.1497412

(Ihme 2005) Ihme, T. & Abrahamsson, P.; (2005) Agile Architecting: The Use of Architectural Patterns in Mobile Java Applications. *International Journal of Agile Manufacturing.* vol. 8 (2). pp. 97-112.

(Janes 2012) Janes, A.A.; & Succi, G.; (2012) The dark side of agile software development. In *Proceedings of the ACM international symposium on new ideas, new paradigms, and reflections on programming and software (ONWARD '12).* pp. 215-228.ACM. doi:10.1145/2384592.2384612

(Jeong 2008) Jeong, Y. J.; Lee, J. H.; & Shin, G. S.; (2008) Development process of mobile application SW based on Agile methodology. In *Proceedings of the 10th International Conference on Advanced Communication Technology, 2008 (ICACT 2008).* vol. 1. pp. 362-366. IEEE. doi:10.1109/ICACT.2008.4493779

(Kamthan 2008) Kamthan, P.; (2008) Towards high-quality mobile applications by a systematic integration of patterns. *Journal of Mobile Multimedia.* vol. 4 (3), pp. 165-184. Rinton Press.

(Kim 2011) Kim, W.Y.; & Park, S.G.; (2011) The 4-tier design pattern for the development of an Android application. In *Proceedings of the Third international conference on Future Generation Information Technology (FGIT'11).* Lecture Notes in Computer Science vol. 7105. pp. 196-203. Springer-Verlag Berlin. doi:10.1007/978-3-642-27142-7_23

(La 2011) La, H. J.; Lee, H. J.; & Kim, S. D.; (2011) An efficiency-centric design methodology for mobile application architectures. In *Proceedings of the 2011 IEEE 7th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob).* pp. 272-279. IEEE. doi:10.1109/WiMOB.2011.6085388

(Liu 2010) Liu, Z.; Gao, X.; & Long, X.; (2010) Adaptive random testing of mobile application In *Proceedings of the 2010 2nd International Conference on Computer Engineering and Technology (ICCET).* vol. 2. pp. 297-301. IEEE. doi:10.1109/ICCET.2010.5485442

(Mantoro 2009) Mantoro, T.; (2009) Metrics evaluation for context-aware computing. In *Proceedings of the 7th International Conference on Advances in Mobile Computing and Multimedia (MoMM '09).* pp. 574-578. ACM. doi:10.1145/1821748.1821859

(Marinho 2012) Marinho, E. H., & Resende, R. F.; (2012) Quality factors in development best practices for mobile applications. In P*roceedings of the 12th international conference on Computational Science and Its Applications (ICCSA'12).* vol. IV. pp. 632-645. Springer-Verlag Berlin / Heidelberg. doi:10.1007/978-3-642-31128-4_47

[Marko 2012] Marko, K.; [2012]. Application development in the age of mobility. *Information Week.* no. 1341. UBM Tech. ISSN 8750-6874.

[Muccini 2012] Muccini, H.; Di Francesco, A.; & Esposito, P.; [2012] Software testing of mobile applications: Challenges and future research directions. In *Proceedings of the 7th International Workshop on Automation of Software Test (AST).* IEEE. pp. 29-35. doi:10.1109/IWAST.2012.6228987

[Nosseir 2012] Nosseir, A.; Flood, D.; Harrison, R.; & Ibrahim, O.; [2012] Mobile development process spiral. In *Proceedings of the 2012 Seventh International Conference on Computer Engineering & Systems (ICCES).* pp. 281-286. IEEE. doi:10.1109/ICCES.2012.6408529

[Pandi 2013] Pandi, K.; & Charaf, H.; [2013] Mobile performance metrics for resource management. In *Proceedings of the International Conference on System Science and Engineering.* IEEE. pp. 329-333. doi: 10.1109/ICSSE.2013.6614686

[Pikkarainen 2005] Pikkarainen, M., Salo, O., & Still, J.; [2005] Deploying agile practices in organizations: A case study, In *Proceedings of the European Software Process Improvement and Innovation Conference (EuroSPI'05).* Lecture Notes in Computer Science, pp. 16-27. Springer Verlag Heidelberg. doi: 10.1007/11586012 3

[Rahimian 2008] Rahimian, V.; & Ramsin, R.; [2008] Designing an Agile methodology for mobile software development: A hybrid method engineering approach. In *Proceedings of the Second International Conference on Research Challenges in Information Science, 2008 (RCIS 2008).* pp. 337-342. IEEE. doi:10.1109/RCIS.2008.4632123

[Rajapakse 2008] Rajapakse, D.C.; [2008] *Fragmentation of mobile applications.* National University of Singapore Available online: http://www.comp.nus.edu.sg/~damithch/df/device-fragmentation.htm [Accessed on January 20th, 2013].

[Ryan 2005] Ryan C.; & Rossi, P.; [2005] Software, performance and resource utilisation metrics for context-aware mobile applications. In *Proceedings of the 11th IEEE International Software Metrics Symposium (METRICS '05).* pp. 10. IEEE Computer Society. doi:10.1109/METRICS.2005.44

(Sa 2008) de Sá, M.; & Carriço, L; (2008) Lessons from early stages design of mobile applications. In *Proceedings of the 10th International Conference on Human Computer Interaction with Mobile Devices and Services (MobileHCI '08).* pp. 127-136. ACM. doi:10.1145/1409240.1409255

(Scharff 2010) Scharff, C.; & Verma, R.; (2010) Scrum to support mobile application development projects in a just-in-time learning context. In *Proceedings of the 2010 ICSE Workshop on Cooperative and Human Aspects of Software Engineering (CHASE '10)* pp. 25-31. ACM. doi:10.1145/1833310.1833315

(Sokolova 2013) Sokolova, K.; Lemercier, M.; & Garcia, L.; (2013) Android passive MVC: a novel architecture model for the Android application development. In *Proceedings of the Fifth International Conference on Pervasive Patterns and Applications (PATTERNS'13).* pp 7-12. IARIA.

(Spriestersbach 2004) Spriestersbach, A.; & Springer, T.; (2004) Quality attributes in mobile web application development. In *Proceedings of the 5th Int. Conf. on Product Focused Software Process Improvement (PROFES'04).* Lecture Notes in Computer Science. Vol. 3009. pp. 120-130. Springer. doi:10.1007/978-3-540-24659-6_9

(UTI 2011) Unified Test Initiative (2011) *Best practice guidelines for developing quality mobile applications.* Available online: http://www.unifiedtestinginitiative.org/files/uti_best_practices_v1_final.pdf (Accessed on September 25th, 2013).

(W3C 2010) W3C Consortium (2010) *Mobile web application best practices.* Available online: http://www.w3.org/TR/mwabp (Accessed on August 1st, 2012).

(Wangenheim 2011) Wangenheim C.G.; & Salazar, L.; (2011) *Mobile software development survey-response summary.* Federal University of Santa Catarina, Brazil. Available online: http://www.gqs.ufsc.br/wp-content/uploads/2011/12/GQS-Workingpaper-002-2011-E-v10.pdf (Accessed on January 20th, 2013).

(Wasserman, 2010) Wasserman, A.I.; (2010) Software engineering issues for mobile application development. In *Proceedings of the FSE/SDP workshop on future of Software Engineering research (FoSER '10).* pp. 397-400. ACM. doi:10.1145/1882362.1882443

Chapter 4:

Quality Attributes of Mobile Applications

Mobile software products are easily accessible to millions of users thanks to the Internet and the mobile application stores. This easiness raises an important concern about trusting the quality of a product to be installed and used on a personal device. Through this work, we have introduced the idea that, to evaluate the quality of a mobile application, it is necessary to have reference points that define themselves the concept of quality. In this way, users and developers will have to take into account uniform characteristics that enable an objective and quantitative product analysis.

The concept of quality has been extensively discussed in specialized literature, but the definition of this term may vary depending on the context or business area in which it is evaluated. Previous chapters have covered different aspects of quality in the Software Engineering domain. Since the concept of quality may be rather subjective, it is hard to provide a definition that holds in all the cases. Let us consider three definitions of quality:

- Conformance to requirements (Crosby 1979)
- Fitness of use (Juran 1951)
- Efficient production of the quality that the market expects (Deming 1986)

With reference to these three definitions, we investigated existing sources of software assessment criteria to provide us with a clear idea of how an application could be considered:

i. in conformance with the requirements,

ii. fit to be utilized, and

iii. efficiently produced to meet what the market expects.

The most suitable source to clarify these conditions are the mobile application stores (app stores), thanks to the deep penetration, influence and impact that they have hosting and disseminating mobile software products. Application stores set up the minimum quality requirements that an app should meet concerning users, developers, execution environment and the mobile business model, suiting our compound definition of quality.

To keep up with a standard quality level, major app stores rely on product development policies and publishing guidelines that every product must observe to be included in the market. Development policies and publishing guidelines imply as well an evaluation processes that provides a trustworthy guide for potential requirements that are relevant for the quality of a mobile product, encompassing the development aspects and the end user's expectations.

In this Chapter, we identified the most relevant quality requirements set upon mobile apps by surveying the publishing guidelines of six major application stores. The selected stores cover a wide range of operative platforms that combined give us a high market coverage: Android OS (Google Play, Amazon Appstore and Nook Apps), Apple (iOS App Store), Windows (Windows Phone Store), and RIM (Blackberry World).

The goal of the analysis presented in this Chapter is to identify the quality requirements that are demanded to the mobile software product in the context of real-world mobile application markets. With the results of our survey, in further sections we shall associate the quality requirements from the mobile app stores with standard quality characteristics, and we will deepen our analysis to single out the most relevant ones.

4.1 Application stores as software delivery platform

Application stores respond to the huge demand of end-user software that takes advantage from the full functionality of mobile smart devices like cellphones and tablets. Today, app markets are the premier channel for the dissemination of mobile software products, serving an immense quantity of developers, users and products. This relevance makes app stores a precious source of information to determine trends in software usage and mobile economy (d'Heureuse 2012). Among others, several market processes have driven the growth of mobile application stores (Kimbler 2010):

i. the growing popularity of mobile devices with broadband Internet access,
ii. the increasing adoption of flat-rate charging plans for mobile data services,
iii. the popularization of high-end, internet and multimedia-enabled mobile handsets, and
iv. the opening of several mobile operating systems for third-party software development.

Software Development Kits (SDK) allow developers to create software applications that are capable to exploit comprehensively the capabilities of the mobile device (e.g., hardware features, telephony management, network use, etc.), and to interface with the operating system and other software components. The ability to produce applications that can be installed, utilized and managed in a mobile phone opened a big market opportunity previously restricted to telephone manufacturers and carrier companies.

4.1.1 The App Store business model

In an effort to facilitate and promote the distribution of applications targeted to mobile operating systems, the "App Store Model" (Hammershoj 2010) constitutes an approach in which a mobile software platform offers the necessary infrastructure for developers and users to respectively offer and obtain software products. Developers utilize the app store as a high-profile platform to showcase their products to millions of potential customers, paying an entrance fee and giving to the store a percentage of the profit

generated by each sale. On the other hand, users access an application delivery platform (usually a website or software interface), search for the desired application, pay for it (if required), download it, and install it in their smart device. An app store offers comprehensive product marketing capabilities, including (Goul 2012):

i. a software distribution model with deployment, installation, updating, deletion or blocking functionalities;

ii. strong support for policy management;

iii. facilities for inventory management, classification, organization, browsing;

iv. security management capabilities (e.g., authentication, encryption, etc.), and

v. service management and analysis functionality.

The App Store Model was popularized by Apple through the iOS App Store for the iPod/iPhone product families. Since then, this model has been the trendsetter for similar software delivery platforms that manage the distribution of mobile software applications in the mainstream. As of 2013, all major mobile operating platforms count on an app store: iPhone has the iOS App Store, Android OS has Google Play, Windows Mobile has the Windows Phone Store, and so on. Likewise, the App Store Model has been applied to other kind of software distribution not necessarily mobile. As an example, the Google Chrome Web Store[7] is a desktop-based application store that specializes on the distribution of browser-enabled applications.

The openness of the operating platform allows the analysis of the App Store Model in two different classes of platforms, depending on the centralization strategy (Gonçalves 2010):

- System integrator platform: all the assets related to the platform and to customer's ownership are in the hands of the platform owner. For instance, the iOS, Windows Phone and Blackberry product families are bounded to a single official software distribution platform (iOS App Store, Windows Marketplace and Blackberry World, respectively).

[7] https://chrome.google.com/webstore

- Enabler platform: the platform owner controls most of the assets involved in the service provision, but several customer relationship capabilities remain open to third-parties. For example, Android OS offers an official distribution market (Google Play) but also welcomes other application stores like Amazon Appstore.

App stores complement the sale of the smart device by giving the possibility of enriching it with software applications. This takes the App Store Model into a business model of higher scope: selling a mobile device is not anymore a stand-alone transaction; instead, it is part of the sale of a complete device-driven ecosystem that consists of the hardware device, its operating system, an application store, and the apps themselves (Rao 2011). By extension, the quality of the products offered by the app market impact the quality of the ecosystem as a whole.

Summarizing, a mature app store model implies the consolidation of a technology-focused environment composed by a product manufacturer (mobile device), a platform provider (operating system), a service intermediary (software delivery engine), software developers and customers.

The App Store Model brings forth potential advantages and different implications for the stakeholders of the mobile software market, namely product developers, software developers and final customers (Tuunainen 2011) (Cortimiglia 2011):

- For product developers (device manufacturers, OS providers), the App Store Model represents an efficient vehicle to consolidate their solutions through powerful, attractive devices managed by proficient operative systems. These solutions will be preferred both by software developers to create and disseminate their products, and by end-users to fulfill their needs. App stores consolidate the sale of the device by keeping a rich and attractive variety of software applications tied to the platform.
- For software developers, using app stores is highly attractive in terms of product exposure, visibility and impact. Moreover, the App Store Model provides many options for generating revenues. In the most straightforward model, developers charge an amount for the app (assigning a fixed price for

downloading the product from the app store). For charged apps, authors retain around 70% of the total revenue, however, pricing an application is not a simple decision: charging may have a negative impact, especially in an environment where free products are very common and highly available. Other means to generate revenues are to sell premium contents, sell advertisement spaces to third parties, allow in-app purchasing, or apply other monetization techniques.

- For end-users, the App Store Model represents a huge source of products in a highly competitive environment. The user's preference represents a very influential market driver, as it consolidates successful operating platforms and dominant application marketplaces. Popular operating platforms with big application markets are most appealing for developers, since their products will reach a wider range of potential customers; in the same sense, large application markets attract a bigger number of users since the variety of products to choose from is larger and richer.

In summary, a popular mobile ecosystem composed by a solid operating system with a powerful software delivery platform is most appealing for developers since their products will have a better chance to reach a wider range of potential customers. In the same way, it will be more attractive for end-users since the variety of products to choose from is larger and richer, and the high competitiveness contributes to keep the prices low.

4.1.2 Software quality assurance in mobile application markets

Mobile-specific software applications bear the big challenge of performing satisfactorily in a heterogeneous and resource-limited environment that demands high availability, efficient performance and short response time, while delivering value to the end user. Although the need of high quality can be minimal under certain viewpoints, in certain business areas (for instance, Security or Defense Sectors) high quality is an imperative, since users rely on communication devices for strategic, safe-critical purposes. For these sectors, the lack of quality represents a serious concern, as the applicability of

traditional software quality strategies may fall short since they do not consider the conditions and particularities of the mobile environment (Corral 2013).

The quality of the mobile software product is normally regulated by market policies and is judged by customer's reviews and ratings. Mobile app stores pose several requirements (i.e., publishing policies) that a product must meet to be included in their catalog listings. Under this rationale, the publishing guidelines prescribed by app stores represent the standard quality level for a mobile app to enter into service. Publishing policies are highly influential for software developers, since the lack of compliance may cause the rejection or exclusion of the product from the app store. Typically, publishing guidelines comprise requirements related to the operation of target device, application content, app functionality, user experience, and several others. In consequence, we can consider publishing requirements as comprehensive and well-settled quality expectations. Additionally, as per publishing guidelines, developers shall cover a number of conditions not necessarily seen as software quality requirements: for instance, the app must have several resources like illustrative pictures, textual descriptions, etc. When a developer wants to promote a product in an app store, he is first required to submit the application to a review phase in which the product is evaluated against the publishing guidelines; if the application is compliant, then it is showcased in the storefront; otherwise it is rejected and the feedback is sent to the submitter.

In addition, the product-oriented body of research described in Chapter 3 suggests the need of identifying relevant quality attributes of mobile applications, and to customize product quality methodologies to suit the needs of the mobile environment. This effort should cover different areas: development practices, testing requirements, product metrics, etc. Our effort picks up from this approach to continue the development of a quality strategy suitable to evaluate the quality of modern mobile apps.

We need to identify which are the requirements that drive the quality of the mobile software product from a market-aware perspective, establish these requirements as high-level goals and introduce the means to assure the fulfillment of such goals

systematically. Covering this process is not a simple endeavor, given the complexity of the mobile environment, the variety of quality drivers, and the diversity of quality criteria found across various application markets. In consequence, we need to set a strategy to discover the requirements that shape what a mobile application should look for, and then to organize them in a standardized way. This will allow us to outline a software quality model that can be applied to assess the quality of any mobile software application. Our methodology is organized in the following stages:

i. Extracting the most important quality requirements from major application stores.

ii. Ordering, summarizing and classifying these quality requirements.

iii. Studying standard-based quality characteristics as a mechanism to assure the fulfillment of the quality requirements from the analyzed application stores.

iv. Constructing an association between standard-based quality characteristics and the quality requirements from the analyzed application stores.

v. Discussing and ranking the most relevant software quality characteristics as result of the association.

vi. Composing a software quality model for mobile application markets.

The detailed description of the six stages constitutes the rest of this Chapter.

4.2 Software quality requirements from mobile application stores

To start our analysis, we retrieved the publishing guidelines from six major application stores, selected to cover comprehensively the major mobile operating platforms and the most important distribution channels. Together, the analyzed platforms cover nearly the 94% of the global smartphone market share (IDC 2013). The publishing guidelines can be retrieved from each application store website:

- **Android OS:** Google Play[8,9], Amazon Appstore [10], Nook Apps [11],
- **Apple iOS:** iOS App Store[12],
- **Microsoft Windows Phone:** Windows Phone Store[13], and
- **RIM Blackberry OS:** Blackberry World[14].

Our analysis is based on the contents of the publishing guidelines updated as of May, 2013. We acknowledge that the application markets reserve the right to revise and change their policies, which may bring as a consequence that some of the analyzed concepts or policies may be subject to change.

First, we focused on analyzing the app store policies to identify all requirements that do not focus on the quality of the software product; for example, those concerning payments, revenues, and the terms and conditions under which developers should use the app store. Since none of these clauses represent characteristics intrinsic to the software product, they were not included in our review. Then, once we set aside the non-product characteristics, we studied the rest of the requirements to ensure that they can be categorized as development guidelines, that is, requirements that can be put on practice while analyzing, designing, coding and testing the application.

To organize our work, we assumed Google's Android development best practices as a baseline to classify the quality requirements. We adopted therefore the classification shown in the Android website[15] as of November of 2012 to organize families of requirements in the different app stores:

[8] http://developer.android.com/guide/practices/index.html

[9] https://play.google.com/about/android-developer-policies.html

[10] https://developer.amazon.com/help/faq.html

[11] https://nookdeveloper.zendesk.com/entries/21247706-nook-apps-developer-policy

[12] https://developer.apple.com/appstore/resources/approval/guidelines.html

[13] http://msdn.microsoft.com/en-US/library/windowsphone/develop/hh184843(v=vs.105).aspx

[14] http://developer.blackberry.com/java/documentation/bp_intro_1984355_11.html

[15] http://developer.android.com/training/index.html

- *Performance:* The ability of the app to execute effectively and efficiently according to the user's needs and device's capabilities.
- *Responsiveness:* The ability of the app to provide a response to the user within a determined amount of time.
- *Seamlessness:* The ability of the app to avoid problems of applications interacting with other applications or with the system.
- *Functionality:* The ability of the application to perform its function as required and intended.
- *Energy Management:* The ability of the app to utilize correctly the available energy resources.
- *Security:* The ability of the app to safeguard the integrity of the user and the execution target.
- *Visual Design and Content:* The ability of the app to display the required visual aspect and contents.

Given the constant evolution of the mobile operating systems and target platforms during the development of this work, the classification above could be subject to changes; however, we kept it as it is since each family of requirements describes efficiently sets of expected traits of mobile apps whose importance certainly holds until the release of this thesis report.

4.2.1 Extracting quality requirements from mobile application stores

Extracting the quality requirements from the publishing guidelines implies to execute a survey in several steps and diverse iterations:

i. reading and understanding the requirements,

ii. isolating the requirements that are relevant to the software product,

iii. grouping them by common characteristics according to the taxonomy previously discussed, and

iv. dismissing the business-specific and environment-specific requirements, since they cannot be compared with requirements from external sources.

The survey was done individually for each publishing policy. After that, we conducted a comparative analysis reviewing the extracted requirements to minimize overlapping or duplication (for instance, requirements expressed in different words but that refer to the same concept, requirements classified under different categories, etc.) As consequence of this exercise, some requirements were slightly rephrased, always preserving the core principle originally expressed.

Table 4 illustrates the most relevant quality requirements extracted from the analyzed publishing policies, after they were classified and organized. The bullet list in each cell intends to present a condensed review of several points that may be explained in more than one requirement in the app market guideline. Finally, if the app store publishing policy does not include any requirement related to a specific category, such category was labeled as "Not included".

Table 4. Quality requirements in major application stores

Characteristic	Google Play	Amazon Appstore	Nook Apps	iOS App Store	Windows Phone Store	Blackberry World
Performance	• Optimize Code • Perform in compliance with device capabilities • Sustain network slowness	• *Not Included*	• Perform in compliance with device capabilities	• *Not Included*	• Perform in compliance with device capabilities	• Sustain network slowness Minimize data usage • Optimize Code
Responsiveness	• Less than 200m seconds for app response	• *Not Included*	• Less than 5 seconds to launch • In-progress indicators to know the status of the app	• *Not Included*	• Less than 5 seconds to launch • App must be responsive 20 seconds after launch	• Keep user informed about the status of the app • Use contextual menus
Seamlessness	• No crashes • Reduce input errors	• *Not Included*	• No crashes • Sessions should be stable	• No crashes • No bugs • No reboots on installations	• No crashes • Sessions should be stable	• Improve the user experience • Reduce input errors • High app scalability
Functionality	• Phone functionality (telephony/SMS) must be of the highest priority	• Perform as advertised • Not to disable critical hardware	• No "lite" applications • Not to disable critical hardware • Provide added value (e.g., no duplicates, useless)	• No hidden features • Perform as advertised • Provide added value (e.g., no duplicates, useless) • No "beta" applications	• Perform as advertised • Assure functionality of device's buttons • Phone functionality (telephony/SMS) must be of the highest priority	• *Not Included*
Energy Management	• Avoid battery drains	• *Not Included*	• Avoid battery drains • Avoid stay awake	• Avoid battery drains	• Avoid battery drains • Follow best practices for	• Avoid battery drains

Characteristic	Google Play	Amazon Appstore	Nook Apps	iOS App Store	Windows Phone Store	Blackberry World
					energy efficiency	
Security	• Permission management • Credential management • Controlled use of external storage	• Avoid putting customer data at risk. • Utilize hardware with official APIs	• Controlled use of external storage • Utilize hardware with official APIs.	• No damages to devices • Avoid putting customer data at risk. • Utilize hardware with official APIs.	• Avoid putting customer data at risk. • No damages to devices • Follow a privacy policy	• Credential management • Encryption • Permission management • Avoid putting customer data at risk.
Visual Design	• Icon guidelines • Support resolutions, icons, size and distributions • Follow Google Design guidelines	• Icon guidelines • Support resolutions, icons, size and distributions.	• Support resolutions, icons, size and distributions. • Icon guidelines	• Icon guidelines • Follow Apple iOS Human Interface Guidelines • Support resolutions, icons, size and distributions. • No excessive space for ads	• Advertisement must comply the Creative Acceptance Policy • Follow Windows Graphic Guidelines	• Follow Blackberry Graphic Guidelines • Utilize the official GUI API
Content	• No offensive content (e.g., pornography, violence, etc.)	• No offensive content • Not to promote	• No offensive content • No advertisement	• Package size limit • No offensive content • No promoting illegal activities	• Disclose package size • No offensive content • Not to promote	• Respect content ratings

4.2.2 Comparative analysis

Once we summarized and classified the requirements from the app store guidelines, we analyzed each group of requirements (that is, each individual cell from Table 4) to extract commonalities and eliminate duplicates. After the list was consolidated, we calculated the frequency count for each requirement to see how many app stores consider it as part of their publishing conditions. The results are shown in Table 5.

Table 5. Summary and frequency of quality characteristics for major mobile application stores

	Google Play	Amazon Appstore	Nook Apps	iOS App Store	Windows Phone	Blackberry World	Frequency
1. Performance							
1.1 Optimize Code	x					x	2
1.2 Performance in compliance with device capabilities	x		x		x		3
1.3 Sustain network slowness	x					x	2
1.4 Minimize data usage						x	1
2. Responsiveness							
2.1 Fixed seconds for app response/launch	x		x		x		3
2.2 Indicators to know the status of the app			x			x	2
2.3 Use of contextual menus						x	1
3. Seamlessness							
3.1 No crashes	x		x	x	x		4
3.2 Reduce input errors	x					x	2
3.3 No bugs				x			1
3.4 No reboots				x			1
3.5 Assure stable			x		x		2

	Google Play	Amazon Appstore	Nook Apps	iOS App Store	Windows Phone	Blackberry World	Frequency
sessions							
3.6 Improve user experience						x	1
3.7 High app scalability						x	1
4. Functionality							
4.1 App should perform as advertised		x		x	x		3
4.2 No hidden features				x			1
4.3 Provision of added value			x	x			2
4.4 No "beta/lite" applications			x	x			2
4.5 Full functionality of device's hardware					x		1
4.6 Highest priority on phone functionality	x		x		x		3
5. Energy management							
5.1 Avoid battery drains	x		x	x	x	x	5
5.2 Avoid stay awake			x				1
5.3 Best practices for energy efficiency					x		1
6. Security							
6.1 Permission management	x					x	2
6.2 Credential management	x					x	2
6.3 Controlled use of external storage	x		x				2
6.4 Not to jeopardize customer data		x		x	x	x	4
6.5 Utilize hardware with official APIs		x	x	x			3
6.6 No damages to devices				x	x		2
6.7 Follow a privacy policy					x		1

	Google Play	Amazon Appstore	Nook Apps	iOS App Store	Windows Phone	Blackberry World	Frequency
6.8 Encryption						x	1
7. Visual Design							
7.1 Icon guidelines	x	x	x	x			4
7.2 Support resolutions, icons, sizes	x	x	x	x			4
7.3 Follow design guidelines	x			x	x	x	4
7.4 Ads in compliance with policy				x	x		2
7.5 Utilization of an official GUI API						x	1
8. Content							
8.1 No offensive content	x	x	x	x	x		5
8.2 Not to promote illegal activities	x	x		x	x		4
8.3 No malware	x				x		2
8.4 No advertisement			x				1
8.5 No reference to other stores			x	x			2
8.6 Package size limit				x	x		2
8.7 Respect content ratings				x		x	2
8.8 Region-specific restrictions					x		1

4.3 Discussion

The summary of quality requirements in Table 5 represents the preliminary set of desired traits or required qualities as expressed by major mobile app stores. The frequency count calculated for each requirement is used to outline what requirements are important for a larger range of stores, and it gives an initial

insight for the analysis conducted in further steps of our study, described in detail in Chapter 5.

We can observe that the quality requirements from the surveyed storefronts revealed great commonalities; they concentrate on elements that affect the user experience, suggesting that distributors are concerned about guaranteeing applications that ensure customer's satisfaction by the seamless execution of the product and the proper exploitation of the capabilities of the device. In addition, app stores are focused particularly on assuring that the content of the applications offered do not break the law at any extent, and always respect the store's brand name and identity. Nonetheless, it is required to conduct a deeper analysis to determine in a justified manner what are the most important quality requirements for mobile software products in the context of mobile application stores.

In this Chapter, we reviewed and compared the publishing guidelines of six of the major mobile app stores that serve the most important mobile operating platforms. We extracted the product quality requirements from their publishing policies and organized them in categories to find commonalities, differences and key factors. Our survey shows a number of commonalities among different app stores; however, further analysis is required to describe what are the specific concerns and to focus on quality assurance throughout the major mobile app stores.

4.4 Chapter bibliography

(Corral 2013) Corral, L.; Sillitti, A.; & Succi, G.; (2013) Agile software development processes for mobile systems: Accomplishment, evidence and evolution. In *Proceedings of the 10th International Conference on Mobile Web Information Systems (MobiWIS 2013).* Lecture Notes in Computer Science, vol. 8093. pp. 90-106. Springer-Verlag Berlin / Heidelberg. doi:10.1007/978-3-642-40276-0_8

(Cortimiglia 2011) Cortimiglia, M.N.; Ghezzi, A.; & Renga, F.; (2011). Mobile applications and their delivery platforms. *IT Professional.* Vol. 13 (5), pp. 51-56. IEEE Computer Society. doi:10.1109/MITP.2011.84

(Crosby 1979) Crosby, P. B.; (1979). *Quality is free.* McGraw-Hill. ISBN: 978-0451621290

(Deming 1986) Deming, W.E.; (1986). *Out of the crisis.* MIT Press. ISBN: 978-0262541152

(d'Heureuse 2012) d'Heureuse, N.; Huici, F.; Arumaithurai, M.; Ahmed, M.; Papagiannaki, K.; & Niccolini, S.; (2012). What's app?: a wide-scale measurement study of smart phone markets. *ACM SIGMOBILE Mobile Computing and Communications Review.* vol. 16 (2), pp. 16-27. ACM. doi:10.1145/2396756.2396759

(Goncalves 2010) Gonçalves, V.; Walravens, N.; & Ballon, P. (2010) How about an app store? Enablers and constraints in platform strategies for mobile network operators. In *Proceedings of the 2010 Ninth International Conference on Mobile Business and 2010 Ninth Global Mobility Roundtable (ICMB-GMR).* pp.66-73. IEEE Computer Society. doi:10.1109/ICMB-GMR.2010.41

(Goul 2012) Goul, M.; Marjanovic, O.; Baxley, S.; & Vizecky, K.; (2012) Managing the enterprise business intelligence app store: Sentiment analysis supported requirements engineering. In *Proceedings of the 2012 45th Hawaii International Conference on System Science (HICSS).* pp. 4168-4177. IEEE Computer Society. doi:10.1109/HICSS.2012.421

(Hammershoj 2010) Hammershoj, A.; Sapuppo, A.; & Tadayoni, R.; (2010) Challenges for mobile application development. In *Proceedings of the 2010 14th International Conference on Intelligence in Next Generation Networks (ICIN).* pp.1-8. IEEE. doi:10.1109/ICIN.2010.5640893

(IDC 2013) International Data Corporation (2013) *Apple cedes market share in smartphone operating system market.* Press Release. Available online: http://www.idc.com/getdoc.jsp?containerId=prUS24257413 (Accessed on September 7th, 2013.).

(Juran 1951) Juran, J.M.; (1951). *Quality Control Handbook.* McGraw-Hill. ISBN: 978-0070331761

(Kimbler 2010) Kimbler, K.; (2010) App store strategies for service providers. In *Proceedings of the 2010 14th International Conference on Intelligence in Next Generation Networks (ICIN).* pp.1-5. IEEE. doi:10.1109/ICIN.2010.5640947

(Rao 2011) Rao, B.; & Jimenez, B.; (2011) A comparative analysis of digital innovation ecosystems. In *Proceedings of Technology Management in the Energy Smart World (PICMET '11).* pp. 1-12. IEEE.

(Tuunainen 2011) Tuunainen, V.K.; Tuunanen, T.; & Piispanen, J.; (2011) Mobile service platforms: comparing Nokia OVI and Apple App Store with the IISIn model. In *Proceedings of the 2011 Tenth International Conference on Mobile Business (ICMB).* pp. 74-83. IEEE Computer Society. doi:10.1109/ICMB.2011.42

Chapter 5: Defining Mobile Specific Quality Attributes

Knowing the expected quality characteristics of different storefronts poses on developers the need of analyzing how their software development practices may suit most of the distribution requirements. This may facilitate strengthening and organizing their quality practices in a way that a single "master" assurance guideline can help to produce applications that fit the distribution policies of more than one application store. This introduces a discussion on how to relate the quality expectations to measurable quality attributes.

5.1 Defining metrics after standardized quality attributes

In this Chapter we pursue to configure the means to assure the systematic fulfillment of the quality requirements from the mobile application stores. To this end, we assume that the quality requirements from the app stores can be met by ensuring a series of quality characteristics expressed in a standardized way. To do it, it is necessary to identify a relationship between the identified market requirements and a family of suitable standard quality attributes.

We revisited a software product quality standard of the ISO series, already suggested for similar domains (as described in detail in Chapter 3). Our selection was based on the comprehensiveness, level of detail, dissemination, and applicability of the standard.

Other software quality standards were dismissed since they were process oriented, business specific, domain specific, or of very narrow applicability. Previous work concentrated on the implementation of ISO/IEC 9126 in the context of mobile software quality, but given the recent evolution of this quality standard and the release of a new one (ISO/IEC 25010:2011), we selected the latter (a detailed comparison of ISO/IEC 25010 and ISO/IEC 9126 can be found in the Annex A of ISO/IEC 25010). We then analyzed the quality characteristics on ISO/IEC 25010 to identify those applicable to the mobile software product in the context defined by the quality requirements of the surveyed app stores.

ISO/IEC 25010 defines a software quality model that can be analyzed in two orthogonal dimensions: developer's viewpoint (internal and external quality) and customer's viewpoint (quality in use), making it very suitable for the app store approach that considers requirements from both perspectives. For each group, ISO/IEC 25010 structures the concept of software quality in two levels: characteristics and sub-characteristics. A characteristic is a category of quality attributes that can be refined into multiple levels of sub-characteristics. In addition, ISO/IEC 25010 establishes that the characteristics and sub-characteristics may be transferred into software quality attributes. An attribute is an entity that can be measured and verified in the software product by the final application or the intermediate work products (Figure 8).

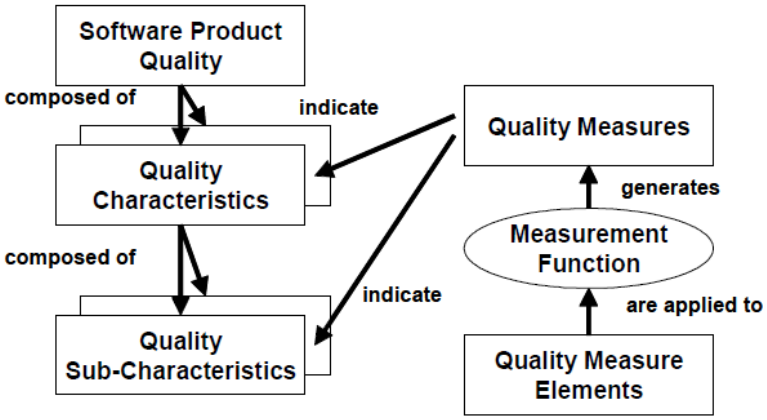


Figure 8: Software product quality reference model from ISO/IEC 25010

It is important to note that only the software quality characteristics and sub-characteristics are defined in the standard; the attributes are not defined because they may vary depending on the final product.

Sections 5.1.1 and 5.1.2 reproduce a detailed description of the quality characteristics and sub-characteristics defined in ISO/IEC 25010, with the goal of identifying opportunities (commonalities, synonyms and direct relationships) to relate the quality requirements from mobile applications stores with the core quality definition of the ISO/IEC 25010 standard.

5.1.1 Internal and External quality: Developer's point of view

According to ISO/IEC 25010, Internal Quality refers to the degree of quality of the software product according to the standard defined in the quality requirement specification (static software measurement). External Quality refers to the degree of quality of the software product according to the standard defined in the quality requirement specification of the system at the integrated level, corresponding to the actual usage (executing software measurement). Internal and External Quality characteristics define "quality" from the development point of view. The standard describes Internal and External Quality through eight characteristics (Figure 9).

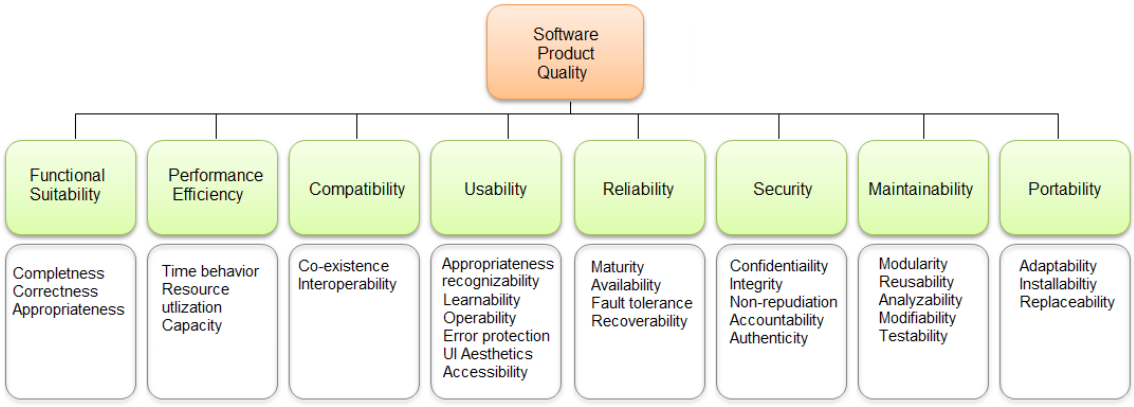


Figure 9: ISO/IEC 25010 product quality model: Internal and External quality characteristics

1. **Functional suitability:** Degree to which a product or system provides functions that meet the stated and implied needs when used under specified conditions.

1.1 *Functional completeness:* Degree to which the set of functions covers all the specified tasks and user objectives.

1.2 *Functional correctness:* Degree to which a product or system provides the correct results with the needed degree of precision.

1.3 *Functional appropriateness:* Degree to which the functions facilitate the accomplishment of specified tasks and objectives.

2. Performance efficiency: Performance relative to the amount of resources used under stated conditions.

2.1 *Time behavior:* Degree to which the response and processing times and throughput rates of a product or system, when performing its functions, meet requirements.

2.2 *Resource utilization:* Degree to which the amounts and types of resources used by a product when performing its functions meet requirements.

2.3 *Capacity:* Degree to which the maximum limits of a product or system parameter meet requirements.

3 Compatibility: Degree to which a product, system or component can exchange information with other products, systems or components, and/or perform its required functions, while sharing the same hardware or software environment.

3.1 *Co-existence:* Degree to which a product can perform its required functions efficiently while sharing a common environment and resources with other products, without detrimental impact on any other product.

3.2 *Interoperability:* Degree to which two or more systems, products or components can exchange information and use the information that has been exchanged.

4. Usability: Degree to which a product or system can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use.

4.1 *Appropriateness recognizability:* Degree to which users can recognize whether a product or system is appropriate for their needs.

4.2 *Learnability:* Degree to which a product or system can be used by specified users to achieve specified goals of learning to use the product or system with effectiveness, efficiency, freedom from risk and satisfaction in a specified context of use.

4.3 *Operability:* Degree to which a product or system has attributes that make it easy to operate and control.

4.4 *User error protection:* Degree to which the system protects users against making errors.

4.5 *User interface aesthetics:* Degree to which the user interface enables pleasing and satisfying interaction for the user.

4.6 *Accessibility:* Degree to which a product or system can be used by people with the widest range of characteristics and capabilities to achieve a specified goal in a specified context of use.

5. Reliability: Degree to which a system, product or component performs specified functions under specified conditions for a specified period of time.

5.1 *Maturity:* Degree to which a system meets needs for reliability under normal operation.

5.2 *Availability:* Degree to which a system, product or component is operational and accessible when required for use.

5.3 *Fault tolerance:* Degree to which a system, product or component operates as intended despite the presence of hardware or software faults.

5.4 *Recoverability:* Degree to which, in the event of an interruption or a failure, a product or system can recover the data directly affected and re-establish the desired state of the system.

6. Security: Degree to which a product or system protects information and data so that persons or other products or systems have the degree of data access appropriate to their types and levels of authorization.

    6.1 *Confidentiality:* Degree to which a product or system ensures that data is accessible only to those authorized to have access.

    6.2 *Integrity:* Degree to which a system, product or component prevents unauthorized access to, or modification of, computer programs or data.

    6.3 *Non-repudiation:* Degree to which actions or events can be proven to have taken place, so that the events or actions cannot be repudiated later (digital signature, etc.)

    6.4 *Accountability:* Degree to which the actions of an entity can be traced uniquely to that entity.

    6.5 *Authenticity:* Degree to which the identity of a subject or resource can be proved to be the one claimed.

7. Maintainability: Degree of effectiveness and efficiency with which a product or system can be modified by the intended maintainers.

    7.1 *Modularity:* Degree to which a system or computer program is composed of discrete components such that a change to one component has minimal impact on other components.

    7.2 *Reusability:* Degree to which an asset can be used in more than one system, or in building other assets.

    7.3 *Analyzability:* Degree of effectiveness and efficiency with which it is possible to assess the impact on a product or system of an intended change to one or more of its parts, or to diagnose a product for deficiencies or causes of failures, or to identify parts to be modified.

    7.4 *Modifiability:* Degree to which a product or system can be effectively and efficiently modified without introducing defects or degrading existing product quality.

7.5 *Testability:* Degree of effectiveness and efficiency with which test criteria can be established for a system, product or component and tests can be performed to determine whether those criteria have been met.

8. Portability: Degree of effectiveness and efficiency with which a system, product or component can be transferred from one hardware, software or other operational or usage environment to another.

    8.1 *Adaptability:* Degree to which a product or system can effectively and efficiently be adapted for different or evolving hardware, software or other operational or usage environments.

    8.2 *Installability:* Degree of effectiveness and efficiency with which a product or system can be successfully installed and/or uninstalled in a specified environment.

    8.3 *Replaceability: D*egree to which a product can be replaced by another specified software product for the same purpose in the same environment.

5.1.2 Quality in Use: Customer's point of view

According to ISO/IEC 25010, Quality in Use is the degree to which a product or system can be used by specific users to meet their needs to achieve specific goals with effectiveness, efficiency, freedom from risk and satisfaction in specific contexts of use.

The quality in use of a system characterizes the impact that the software product has on stakeholders. It is determined by the quality of the software, hardware and operating environment, and the characteristics of the users, tasks and social environment. Quality in Use characteristics are available only when the final product is used in real-world execution conditions, so they represent the point of view of the final customer with respect to the quality of the product. In this perspective, the quality standard describes 5 quality characteristics (Figure 10).

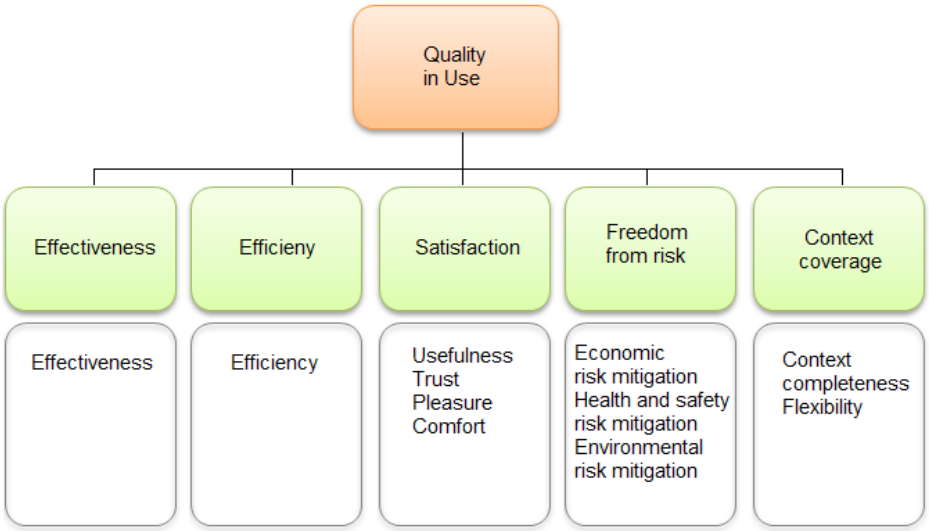


Figure 10: ISO/IEC 25010 product quality model: Quality in Use characteristics

1. **Effectiveness:** Accuracy and completeness with which users achieve specified goals.
2. **Efficiency:** Degree of resources expended in relation to the accuracy and completeness with which users achieve goals.
3. **Satisfaction:** Degree to which user needs are satisfied when a product or system is used in a specified context of use.
    - 3.1 *Usefulness:* Degree to which a user is satisfied with their perceived achievement of programmatic goals, including the results of use and the consequences of use.
    - 3.2 *Trust:* Degree to which a product or system will behave as intended.
    - 3.3 *Pleasure:* Degree to which a user obtains pleasure from fulfilling their personal needs.
    - 3.4 *Comfort:* Degree to which the user is satisfied with physical comfort.
4. **Freedom from risk:** Degree to which a product or system mitigates the potential risk to economic status, human life, health or the environment.

4.1 *Economic risk mitigation:* Degree to which a product or system mitigates the potential risk to financial status, efficient operation, commercial property, reputation or other resources in the intended contexts of use.

4.2 *Health and safety risk mitigation:* Degree to which a product or system mitigates the potential risk to people in the intended contexts of use.

4.3 *Environmental risk mitigation:* Degree to which a product or system mitigates the potential risk to property or the environment in the intended contexts of use.

5. Context coverage: Degree to which a product or system can be used with effectiveness, efficiency, freedom from risk and satisfaction in both specified contexts of use and in contexts beyond those initially explicitly identified.

   5.1 Context completeness: Degree to which a product or system can be used with effectiveness, efficiency, freedom from risk and satisfaction in all the specified contexts of use.

   5.2 Flexibility: Degree to which a product or system can be used with effectiveness, efficiency, freedom from risk and satisfaction in contexts beyond those initially specified in the requirements.

ISO/IEC 25010 does not include a definition of the detailed software quality attributes, leaving them open to suit the needs of each specific software product. Our approach benefits from this flexibility, as it opens the door to adapt this quality model to the requirements from any particular point of view. In this way, we may use the essential software quality characteristics of ISO/IEC 25010 in concomitance with the requirements previously identified on the app markets, with the goal of implementing a mobile-specific quality model on top of a robust software quality standard. The mobile-specific software quality model shall provide a reference point to appraise of mobile applications in a product-based perspective, with respect to the expectations of the application markets, from the point of view of developers and end-users. Eventually, the attributes can be related to metrics, permitting a quantitative way to measure the

quality of the product and deliver relevant information about it, which can be later used to feedback the development and assurance processes (MOETIJ 2011).

5.2 Associating ISO/IEC 25010 with mobile-specific quality requirements

After isolating the quality requirements from mobile app stores and examining the quality characteristics of ISO/IEC 25010, we need to describe a substantiated association between them. To reach this goal, we considered the implementation of principles of customer-driven engineering, which allows us to assure the design quality of the product based on characteristics implicitly and explicitly desired by the customer. This approach may be implemented in the form of the Quality Function Deployment (QFD) (Akao 1994).

The QFD helps to save design and development time, but more importantly it focuses on the satisfaction of end users. The main goals of a QFD are (Mynt 2003):

i. prioritize spoken and unspoken customer wants and needs,
ii. translate these needs into technical characteristics and specifications, and
iii. build and deliver a quality product or service by focusing on customer satisfaction.

In the area of software development, the QFD has been previously regarded as a suitable way to manage and process requirements (Karlsson 1997) (Ramires 2005), (Kivinen 2008), (Xiong 2008) permitting:

i. a better focus on customers and users,
ii. an effective tool for prioritizing and communicating software requirements, and
iii. an efficient way to define and manage non-functional requirements.

As the quality requirements from the different app stores reviewed in the previous Chapter reflect the needs of the customer, the target device, the execution environment,

and the mobile business, a QFD facilitates the redefinition of such needs by means of non-functional requirements, and provides the tools to organize and rank them.

We used a Quality Function Deployment to fulfill the mobile market expectations by translating them into design characteristics. By applying the QFD to this work, we prioritized the customer needs reflected in the quality requirements from app stores, and translated them into the characteristics expressed by the ISO/IEC 25010 quality standard. To conclude, we ranked the results of the QFD to build a quality model to evaluate mobile applications that pursue the satisfaction of the customer through the fulfillment of the requirements established by the major mobile app stores.

5.2.1 Implementation of the Quality Function Deployment

To guarantee the accuracy of the QFD, it was developed in collaboration with a group formed by two university full professors specialized in Software Engineering, two graduate students researching Mobile Software Engineering, a Software Engineering senior specialist from the Italian Army and three staff members of three software companies specialized in the development of mobile applications. We based our methodology on the procedure proposed by Auckland University of Technology (CIRI 2007), which establishes a series of steps to build a QFD that ensures that the customer requirements are accurately translated into relevant quality characteristics.

Step 1. Customer Requirements: The initial step is the elicitation of customer requirements, gathering information about what are the desired characteristics that the product should have. We obtained this information from Table 5, extracting the 44 quality requirements taken from the app stores, and classifying them as the "Demanded Quality".

Step 2. Customer Importance Ratings: It is necessary to rate the importance of each quality requirement (QR) since this number will be used later in a relationship matrix. We ranked the importance from 1 to 6, based on the frequency of each requirement among the six analyzed app stores. Under this criterion, the minimum level of importance is 1, for those requirements that appear only in one application store, and

the maximum level of importance is potentially 6, for those requirements that would be considered by all the six application stores (column "Frequency" from Table 5). We called this value the Demanded Quality Weight (DQW). In addition, we calculated the Demanded Quality Relative Weight (DQRW), which indicates the overall importance of the quality requirement relative to the weights of all other quality requirements. The DQRW is given by the quotient of the Demanded Quality Weight divided by the sum of the weights of all other Demanded Quality Weights (1). To express this value in percentage notation, we multiplied it by 100.

$$DQRW = \frac{DQW}{\sum_{QR=1}^{n} (DQW)} \tag{1}$$

The computed values of Demanded Quality Weight and Demanded Quality Relative Weight are shown in Table 6.

Table 6. Demanded quality weight and demanded quality relative weight

Demanded Quality Requirements (from App Markets)	Demanded Quality Weight	Demanded Quality Relative Weight
1.1 Optimize Code	2	2.08%
1.2 Compliance with device capabilities	3	3.13%
1.3 Sustain network slowness	2	2.08%
1.4 Minimize data usage	1	1.04%
2.1 Fixed seconds for app response/launch	3	3.13%
2.2 Indicators to know the app status	2	2.08%
2.3 Use of contextual menus	1	1.04%
3.1 No crashes	4	4.17%
3.2 Reduce input errors	2	2.08%
3.3 No bugs	1	1.04%
3.4 No reboots	1	1.04%
3.5 Assure stable sessions	2	2.08%
3.6 Improve user experience	1	1.04%
3.7 High app scalability	1	1.04%

Demanded Quality Requirements (from App Markets)	Demanded Quality Weight	Demanded Quality Relative Weight
4.1 App should perform as advertised	3	3.13%
4.2 No hidden features	1	1.04%
4.3 Provision of added value	2	2.08%
4.4 No "beta/lite" applications	2	2.08%
4.5 Full functionality of device's hardware	1	1.04%
4.6 Highest priority on phone functions	3	3.13%
5.1 Avoid battery drains	5	5.21%
5.2 Avoid stay awake	1	1.04%
5.3 Best practices for energy efficiency	1	1.04%
6.1 Permission management	2	2.08%
6.2 Credential management	2	2.08%
6.3 Controlled use of external storage	2	2.08%
6.4 Not to jeopardize customer data	4	4.17%
6.5 Utilize hardware with official APIs	3	3.13%
6.6 No damages to devices	2	2.08%
6.7 Follow a privacy policy	1	1.04%
6.8 Encryption	1	1.04%
7.1 Icon guidelines	4	4.17%
7.2 Support resolutions, icons, sizes	4	4.17%
7.3 Follow design guidelines	4	4.17%
7.4 Ads in compliance with policy	2	2.08%
7.5 Utilization of an official GUI API	1	1.04%
8.1 No offensive content	5	5.21%
8.2 Not to promote illegal activities	4	4.17%
8.3 No malware	2	2.08%
8.4 No advertisement	1	1.04%
8.5 No reference to other stores	2	2.08%
8.6 Package size limit	2	2.08%
8.7 Respect content ratings	2	2.08%
8.8 Region-specific restrictions	1	1.04%

Step 3. Quality Characteristics: The Quality Characteristics describe the product in terms of attributes that may be measured. In our QFD, the Quality Characteristics were

taken from the 42 quality sub-characteristics of ISO/IEC 25010 (Quality sub-characteristics listed in sections 5.1.1 and 5.1.2), since they describe with the highest granularity available the traits that the developer should consider towards assuring the quality of the software product.

Step 4. Relationship Matrix: The relationship matrix determines the interaction between the Demanded Quality and the Quality Characteristics, that is, the relationship between the customer needs and the instruments at hand to fulfill them. In accordance to ISO/IEC 25010, we performed a separate relationship analysis for the Internal/External Quality characteristics and for the Quality in Use characteristics. The QFD requires the calculation of a Relationship Value (RV) that describes the association between the demanded quality requirements and the quality characteristics. We calculated the RV according to the referenced methodology, which recommends scaling the value in three levels:

- 9 for a strong relationship,
- 3 for a moderate relationship,
- 1 for a weak relationship.

A null value represents no relationship. For a better organization, the detailed description of the relationship between the quality requirements from app stores and Internal/External Quality and Quality in Use characteristics is shown in Appendix A.

Step 5. Quality Characteristic Importance Ratings: The association between the Demanded Quality and the Quality Characteristics allows the calculation of parameters that denote the relevance of each quality requirement. First, we obtained the Quality Characteristic Weight (QCW), which indicates the overall importance of the quality characteristic. It is given by the sum of all Relationship Values between each Demanded Quality Requirement and the analyzed Quality Characteristic, multiplied by their corresponding Demanded Quality Relative Weight (2).

$$QCW = \sum_{QC=1}^{n} (RV * DQRW)$$

(2)

Then, we calculated the Quality Characteristic Relative Weight (QCRW), which indicates the weight of the quality characteristic relative to the weights of other quality characteristics. It is obtained by the quotient of the Quality Characteristic Weight divided by the sum of all other Quality Characteristic Weights (3). To express this value in percentage notation, we multiplied it by 100.

$$QCRW = \frac{QCW}{\sum_{QC=1}^{n} (QCW)}$$

(3)

The computed values of the Quality Characteristic Weight and Quality Characteristic Relative Weight are included in Table 7 for the Internal/External quality characteristics and in Table 8 for the Quality in Use characteristics.

Table 7. Internal/External quality sub-characteristic weight and relative weight

Quality Sub-characteristics (from ISO/IEC 25010)	Quality Characteristic Weight	Quality Characteristic Relative Weight
1.1 Functional completeness	176.042	6.18%
1.2 Functional correctness	180.208	6.32%
1.3 Functional appropriateness	234.375	8.22%
2.1 Time behavior	95.833	3.36%
2.2 Resource utilization	162.500	5.70%
2.3 Capacity	100.000	3.51%
3.1 Co-existence	73.958	2.60%
3.2 Interoperability	71.875	2.52%
4.1 Appropriateness recognizability	211.458	7.42%
4.2 Learnability	87.500	3.07%

Quality Sub-characteristics (from ISO/IEC 25010)	Quality Characteristic Weight	Quality Characteristic Relative Weight
4.3 Operability	135.417	4.75%
4.4 User error protection	32.292	1.13%
4.5 User interface aesthetics	159.375	5.59%
4.6 Accessibility	64.583	2.27%
5.1 Maturity	94.792	3.33%
5.2 Availability	32.292	1.13%
5.3 Fault tolerance	26.042	0.91%
5.4 Recoverability	42.708	1.50%
6.1 Confidentiality	121.875	4.28%
6.2 Integrity	127.083	4.46%
6.3 Non-repudiation	228.125	8.00%
6.4 Accountability	175.000	6.11%
6.5 Authenticity	104.167	3.65%
7.1 Modularity	15.625	0.55%
7.2 Reusability	15.625	0.55%
7.3 Analyzability	15.625	0.55%
7.4 Modifiability	15.625	0.55%
7.5 Testability	6.250	0.22%
8.1 Adaptability	15.625	0.55%
8.2 Installability	18.750	0.66%
8.3 Replaceability	9.375	0.33%

Table 8. Quality in Use sub-characteristic weight and relative weight

Quality Sub-characteristics (from ISO/IEC 25010)	Quality Characteristic Weight	Quality Characteristic Relative Weight
1. Effectiveness	211.46	9.54%
2. Efficiency	211.46	9.54%
3.1 Usefulness	225.00	10.15%
3.2 Trust	347.92	15.70%
3.3 Pleasure	257.29	11.61%
3.4 Comfort	222.92	10.06%

Quality Sub-characteristics (from ISO/IEC 25010)	Quality Characteristic Weight	Quality Characteristic Relative Weight
4.1 Economic risk mitigation	190.63	8.60%
4.2 Health and safety risk mitigation	178.13	8.04%
4.3 Environmental risk mitigation	86.46	3.90%
5.1 Context completeness	255.21	11.51%
5.2 Flexibility	30.21	1.36%

5.2.2 Prioritization of the quality characteristics

After the calculation of the relative weights for each quality characteristic, we proceeded to rank them in descending order. In this way, it will be easier to note what are the most important quality characteristics associated to the most relevant quality requirements. Since the relationship analysis was conducted separately, in Figure 11 we show the ranking of Internal/External Quality characteristics, and in Figure 12 the ranking of Quality in Use characteristics.

From the development point of view, it is clear that application markets concentrate on showcasing products that are functional and that meet the expectations of the end user: products shall not present any flaw and shall not lead to any malfunction in the mobile system. Moreover, applications should be highly usable, organized, attractive and aesthetic, and must make a wise use of the device's resources. In a second level of relevance, we noted that application markets have a particular focus in assuring the trustworthiness of the product, in terms of integrity, confidentiality and privacy.

Finally, we observed that app stores are not particularly concerned with the method that developers follow to create a product as long as the result is attractive, safe and flawless for the user and the device. Purely development-oriented characteristics such as modularity, reusability, analyzability, modifiability, testability and others have the lowest relevance (Figure 11).

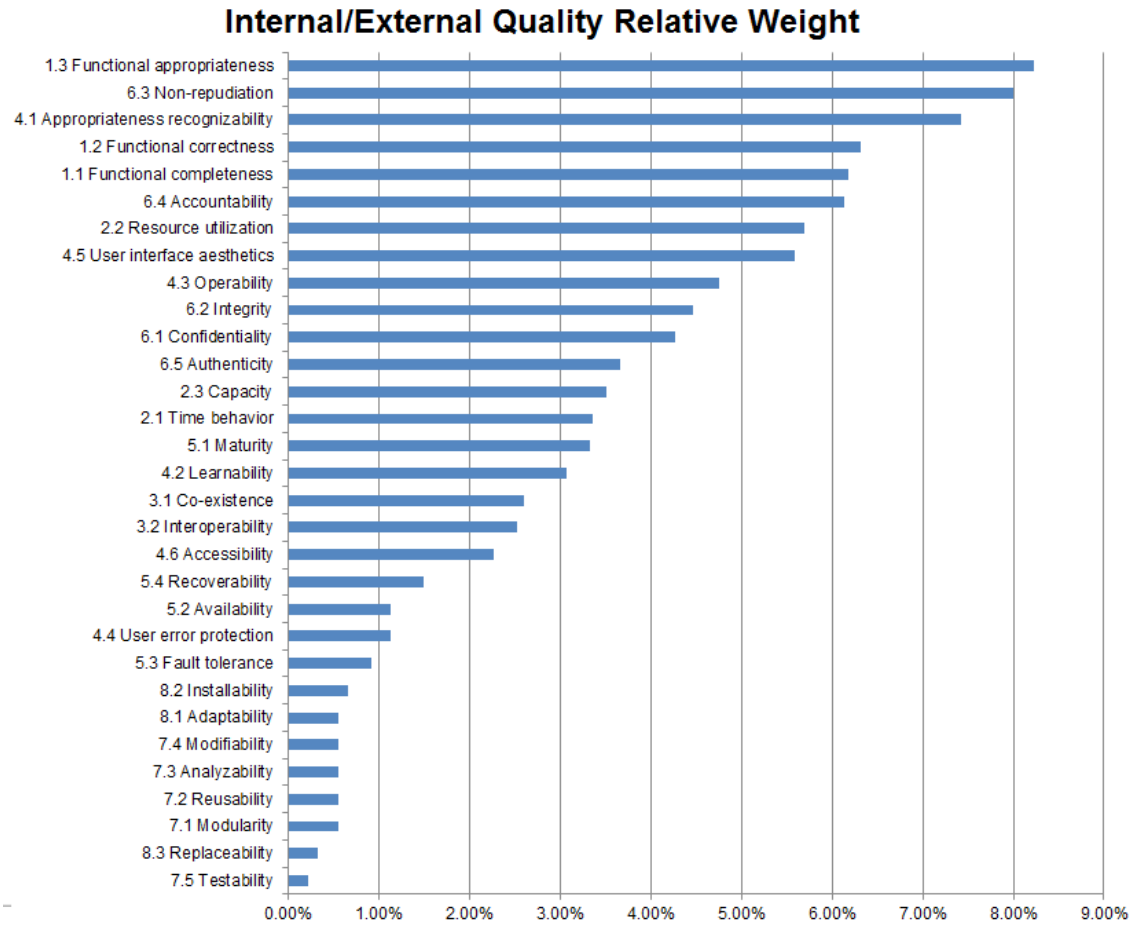


Figure 11: Internal and External quality sub-characteristics ranked by relative weight

Nonetheless, it is important to note that, even though application stores do not explicitly care about the design and implementation practices followed in the development of an app, such design and implementation practices may have a significant impact on characteristics that are highly considered by their quality requirements (further analysis is provided in Chapter 6, with specific examples for responsiveness and energy consumption).

From the user's point of view, it is noticeable that most of the Quality in Use sub-characteristics are relevant to the publishing policies of the app stores. This is not surprising, since these requirements tend to focus on customer satisfaction, denoted directly by the trustworthiness, usefulness, comfort, effectiveness and efficiency of the

mobile application. These traits are related to the ability of the application to perform correctly and help the user to solve a need in a trustworthy, pleasant and gratifying way. With the exception of Flexibility and Environmental Risk Mitigation, all others Quality in Use sub-characteristics are rather relevant to the quality requirements of mobile application stores (Figure 12).

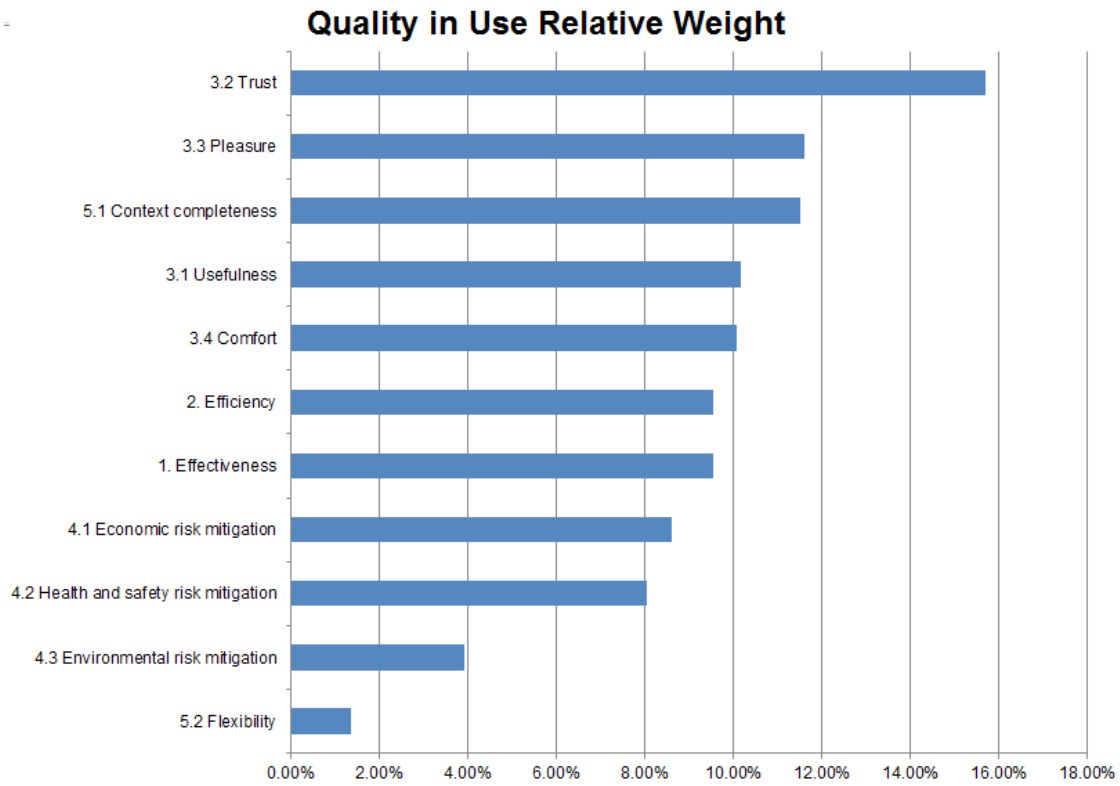


Figure 12: Quality in Use sub-characteristics ranked by relative weight

As a final step, we introduced a software quality model for mobile applications based on the analysis of quality requirements and the quality characteristics that described them better. This quality model intends to assist software developers in evaluating the quality of the mobile software product based on the standard quality characteristics that were noted to suit better the quality requirements of the major mobile app stores.

5.3 Composition of the market-based Mobile App Quality Model

The association between the quality requirements and the quality characteristics that address them allows us to outline a Mobile App Quality Model. The Mobile App Quality

Model instantiates a general-purpose quality model that privileges the specific needs of the mobile product in the context of the mobile application market. This implies that the model should focus on the quality characteristics that deliver more value to the quality requirements of the app stores and dismiss the quality characteristics that were identified as not relevant. The exclusion of the non-relevant quality characteristics permits us to compose a simplified but reliable quality model.

To assess the importance of the quality sub-characteristics, we appraised the value of their Relative Weights, in light of the ranking of Section 5.2.2. To conduct the selection, we defined a threshold, i.e., a minimum Relative Weight (QCRW) that each sub-characteristic must exceed to be included in the final model. Every sub-characteristic with a QCRW lower than the threshold value was dismissed. To guarantee a high level of confidence in the final model, we kept the sub-characteristics that lead to a potential level of accomplishment (confidence) of more than 90% based on the total sum of their Relative Weights. For Internal/External Quality, the threshold was set to 2%, while for Quality in Use the threshold was set to 8%. Following these criteria, 12 Internal/External quality sub-characteristics and 2 Quality in Use sub-characteristics were dismissed, reducing the Mobile App Quality Model to 19 Internal/External quality sub-characteristics and 9 Quality in Use sub-characteristics.

This reduction permitted us to simplify the quality model while keeping a high level of confidence in meeting the requirements of the app markets. The Internal/External Quality characteristics that form the Mobile App Quality Model are shown in Table 9, and the selected Quality in Use characteristics are shown in Table 10.

Table 9. Mobile App Quality Model: selected Internal/External quality sub-characteristics

Internal/External Quality Sub-Characteristics	Relative Weight
1.1 Functional completeness	6.18%
1.2 Functional correctness	6.32%
1.3 Functional appropriateness	8.22%
2.1 Time behavior	3.36%
2.2 Resource utilization	5.70%

Internal/External Quality Sub-Characteristics	Relative Weight
2.3 Capacity	3.51%
3.1 Co-existence	2.60%
3.2 Interoperability	2.52%
4.1 Appropriateness recognizability	7.42%
4.2 Learnability	3.07%
4.3 Operability	4.75%
4.5 User interface aesthetics	5.59%
4.6 Accessibility	2.27%
5.1 Maturity	3.33%
6.1 Confidentiality	4.28%
6.2 Integrity	4.46%
6.3 Non-repudiation	8.00%
6.4 Accountability	6.14%
6.5 Authenticity	3.65%
Sum of Total Weights (Confidence)	91.37%

Table 10. Mobile App Quality Model: selected Quality in Use sub-characteristics

Quality in Use Sub-characteristics	Relative Weight
1. Effectiveness	9.54%
2. Efficiency	9.54%
3.1 Usefulness	10.15%
3.2 Trust	15.70%
3.3 Pleasure	11.61%
3.4 Comfort	10.06%
4.1 Economic risk mitigation	8.60%
4.2 Health and safety risk mitigation	8.04%
5. Context completeness	11.51%
Sum of Total Weights (Confidence)	94.74%

Finally, we organized the quality sub-characteristics consolidating them with their corresponding quality characteristic, following the original structure and organization of ISO/IEC 25010. To maintain consistency with our analysis and the standard itself, we

kept the separation between Internal/External quality characteristics (development-oriented) and the Quality in Use quality characteristics (user-oriented).

Based on the relationship matrix developed while calculating the QFD, we composed a textual definition to provide a better understanding of each one and to explain why the quality characteristic covers the corresponding app store quality requirements.

5.3.1 Mobile Internal and External quality: Developer's point of view

Emulating the corresponding ISO/IEC 25010, the Development Oriented Quality Model refers to the degree of quality of the software product according to the standard defined in the quality requirement specification and the usage during execution, and it is described by several quality characteristics and sub-characteristics (Figure 13):

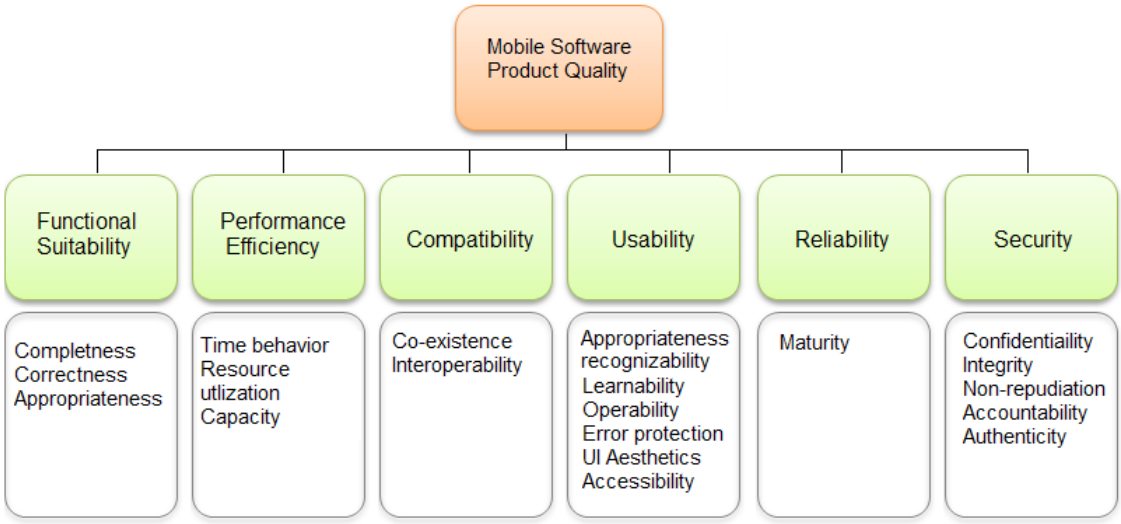


Figure 13: Mobile App Quality Model: development-oriented quality characteristics

1. **Functional suitability:** Degree to which the mobile application provides functions as advertised, meeting the stated and implied needs when used under specified conditions.

    1.1 *Functional completeness:* Degree to which the functions of the mobile application cover all the specified tasks and user objectives.

1.2 *Functional correctness:* Degree to which the mobile provides the correct results with the needed degree of precision and transparency as advertised.

1.3 *Functional appropriateness:* Degree to which the functions of the mobile application facilitate the accomplishment the user's goals, providing added value with respect to similar products.

2. Performance efficiency: Degree to which the mobile application performs with the amount of resources provided by the mobile execution environment.

   2.1 *Time behavior:* Degree to which the launch, processing, response and network connectivity times and throughput rates of the mobile application meet the requirements.

   2.2 *Resource utilization:* Degree to which the amounts and types of device, energy, data and network resources used by the mobile app when performing its functions meet requirements.

   2.3 *Capacity:* Degree to which the mobile application can sustain the necessary processing, network traffic and data management to meet the requirements.

3. Compatibility: Degree to which the mobile application can exchange information with other software components, applications, operative system, device or data networks, and perform its required functions, while sharing the same hardware or software environment.

   3.1 *Co-existence:* Degree to which the mobile application can perform its required functions efficiently while sharing a common environment, software, hardware and energy resources with other products, without detrimental impact on itself or any other product.

   3.2 *Interoperability:* Degree to which two or more mobile applications can exchange information and use the information that has been exchanged.

4. Usability: Degree to which a mobile application can be used by the user to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use.

    4,1 Appropriateness recognizability: Degree to which users can recognize whether the mobile application is appropriate for their needs.

    4.2 Learnability: Degree to which users succeed in learning to use mobile application with effectiveness, efficiency, freedom from risk and satisfaction in a specified context of use.

    4.3 Operability: Degree to which the mobile application has the necessary operative attributes, instruments and controls that make it easy to operate.

    4.4 User interface aesthetics: Degree to which the mobile application's user graphic interface enables pleasing and satisfying interaction with the user, while complying with the visual requirements of the operative platform.

    4.5 Accessibility: Degree to which the mobile application can be used by people with the widest range of characteristics and capabilities to achieve a specified goal in a specified context of use.

5. Reliability: Degree to which the mobile application performs specified functions under specified conditions for a specified period of time.

    5.1 *Maturity:* Degree to which the mobile application meets needs for reliability under normal operations, safeguarding its target environment and user integrity.

6. Security: Degree to which the mobile application protects information and data so that users have the degree of data access appropriate to their types and levels of authorization.

    6.1 *Confidentiality:* Degree to which the mobile application ensures that data is accessible only by those authorized to have access.

6.2 *Integrity:* Degree to which the mobile application operates to prevent unauthorized access to restricted data or execution of unsafe operations.

6.3 *Non-repudiation:* Degree to which actions or events executed by the mobile application cannot be repudiated later because of their functionality, behavior, content, rating, lawfulness and integrity compliance.

6.4 *Accountability:* Degree to which the actions of the mobile application can be traced uniquely to that entity, performing exclusively as advertised and avoiding hidden features.

6.5 *Authenticity:* Degree to which the identity of a subject or resource acting within the mobile application can be proved to be the one claimed.

5.3.2 Quality in Use: Customer's point of view

The Mobile App User-Oriented Quality Model describes the degree to which a product or system can be used by specific users to meet their needs to achieve specific goals with effectiveness, efficiency, freedom from risk and satisfaction in the mobile context of use. It is described by five quality characteristics and sub-characteristics (Figure 14):

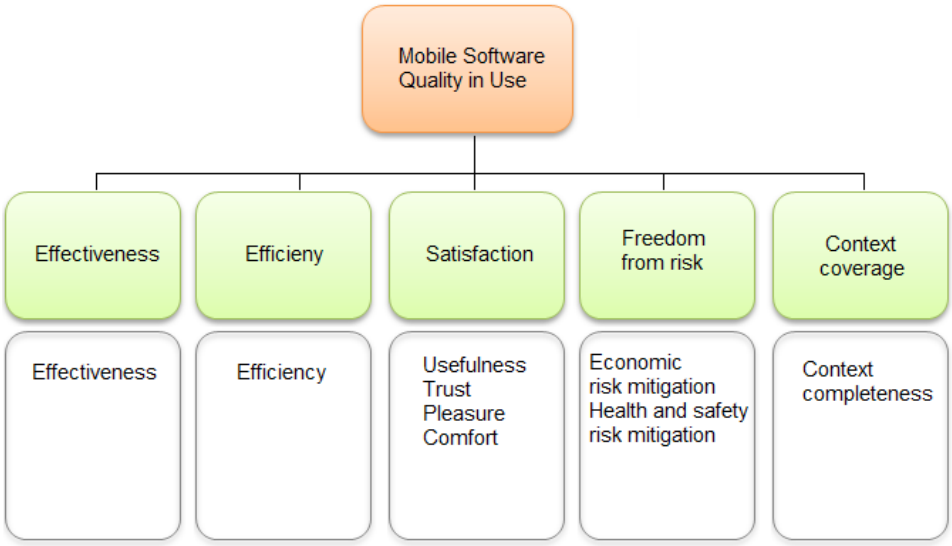


Figure 14: Mobile App Quality Model: user-oriented quality characteristics

1. **Effectiveness:** Degree of accuracy, seamlessness and completeness with which users achieve specified goals by using the mobile application.
2. **Efficiency:** Degree of hardware, software, network and energy resources expended in relation to the accuracy, performance, responsiveness completeness with which users achieve goals by using the mobile application.
3. **Satisfaction:** Degree to which user's needs are satisfied when the mobile application is used in a specified context of use.
    - 3.1 *Usefulness:* Degree to which the mobile application satisfies a user in the perceived achievement of goals, including the results of use and the consequences of use and the provision of added value with respect to competing products.
    - 3.2 *Trust:* Degree to which the mobile application will behave as intended, communicating the execution, status, avoiding interruptions, minimizing user errors, and assuring stable sessions.
    - 3.3 *Pleasure:* Degree to which users obtain pleasure from fulfilling their personal needs, including the provision of appropriate content and an aesthetic operative interface.
    - 3.4 *Comfort:* Degree to which the mobile application satisfies a user with physical comfort, including fostering the user experience, maintaining a harmonic operative interface, and following design guidelines.
4. **Freedom from risk:** Degree to which the mobile application mitigates the potential risk to economic status, user safety, target device safety, or the environment.
    - 4.1 *Economic risk mitigation:* Degree to which the mobile application mitigates the potential risk to financial status, efficient operation, or other resources in the intended contexts of use, including the operation of the device, exploitation of energy resources, utilization of data connectivity, and overall resource usage.

4.2 *Health and safety risk mitigation:* Degree to which the mobile application mitigates the potential risk to people in the intended contexts of use, assuring its behavior, content, rating, lawfulness and overall integrity compliance.

5. Context completeness: Degree to which the mobile application can be used with effectiveness, efficiency, freedom from risk and satisfaction in all the specified contexts of use beyond those explicitly defined.

Although we do not claim that compliance with the Mobile App Quality Model guarantees infallibly the admission of the app into the mobile storefront, we maintain that counting on mobile-specific quality characteristic contributes to a better comprehension of the quality requirements expected by different mobile app markets, and facilitates the definition of parameters to evaluate the quality of the mobile software product with respect to such requirements. Moreover, the total sum of the Relative Weights of the selected quality characteristics denotes a high level of confidence of the Mobile App Quality Model with respect of what is required by the publishing guidelines of the major mobile application stores.

After identifying and prioritizing the most relevant quality characteristics from the major mobile app stores and defining the Mobile App Quality Model, it is necessary to conduct an experiment to implement the model and test its applicability and usefulness in a real-world scenario. The quality characteristics defined by the Mobile App Quality Model should be set down in attributes that can be measured and verified. Then, these attributes should be refined in metrics that provide a quantitative way to measure the quality of the product, by applying them to real mobile software applications.

5.4 Creating mobile-specific quality metrics with a partially-instantiated GQM

Built on top of ISO/IEC 25010, the Mobile App Quality Model does not include a description of the detailed quality attributes, leaving them open to fit the specific needs of the vast spectrum of mobile apps. From these custom quality attributes, one may

define the metrics that help to conduct the quantitative analysis of the goals set by the quality characteristics. Thus, to complete our quality model we provide an instrument to define metrics that suit the measurement needs of the quality characteristics.

The Goal Question Metric (Basili 1994) approach consists of the specification of a measurement system targeting a particular set of issues and a set of rules for the interpretation of the measurement data. The resulting measurement model has three levels:

1. Conceptual level (Goal): A goal is defined for an object, for a variety of reasons, with respect to various models of quality, from various points of view, relative to a particular environment. Objects of measurement are:
    - *Products*: Artifacts, deliverables and documents that are produced during the system life cycle.
    - *Processes*: Software related activities normally associated with time.
    - *Resources*: Items used by processes in order to produce their outputs.
2. Operational level (Question): A set of questions is used to characterize the way the assessment/achievement of a specific goal is going to be performed based on some characterizing model. Questions try to characterize the object of measurement with respect to a selected quality issue and to determine its quality from the selected viewpoint.
3. Quantitative level (Metric): A set of data is associated with every question in order to answer it in a quantitative way. The data can be:
    - *Objective*: If they depend only on the object that is being measured and not on the viewpoint from which they are taken.
    - *Subjective*: If they depend on both the object that is being measured and the viewpoint from which they are taken.

We pursue to know how to relate mobile-specific quality expectations with attributes that may assist developers in the production of mobile apps that suit the requirements imposed by the analyzed application stores, and we presented an initial effort to identify

the most important quality requirements set upon Android mobile apps. This analysis sets the basis to introduce a quality strategy to relate the major quality requirements from mobile apps into measurable attributes, to aid the development and assurance processes so as to guarantee the production and delivery of compliant mobile applications.

The GQM approach provides a method for an organization or project to de**fi**ne goals, re**fi**ne these goals down to the speci**fi**cation of data to be collected, and then analyze and interpret the resulting data with respect to the original goals (Mandic 2010). We propose an Agile implementation of the GQM approach defined in a way to provide an efficient approximation of quality for the domain of mobile software product, considering beforehand the conditions that typically arise in the mobile domain and that impact the quality of the resulting product. By applying it, the time to produce a new effective metric will be shortened.

To maximize the benefit of this approach, we propose to summarize the common requisites of mobile software quality and prepare a GQM template partially filled in part with mobile-specific quality indicators that can be later detailed and customized. In this way, it will be guaranteed that the resulting metrics will be beneficial to keep track of mobile-specific quality drivers (Corral 2013).

In other words, we plan to create a partial instantiation that allows us to establish a core, extensible GQM that can be tailored, depending on the focus of the analysis. The goal will be tied to the high level objective of this work, the analysis of the quality of the product:

> *G.1:* "Analyze the mobile software product for the purpose of evaluating it with respect to the quality, from the view point of developers and customers, in the context of execution environment and application markets".

The questions, on the other hand, can be preliminarily set to survey the characteristics of the application and to evaluate it with respect to an outstanding attribute (X) that relates directly to the issue that the GQM model attempts to solve:

Q.1: "What is the current performance of the application with respect to attribute X?"

Q.2: "Is the current performance of attribute X satisfactory from the viewpoint of the developer and the user?"

Q.3: "Is the current performance of attribute X acceptable from the viewpoint of the application market?"

With respect to the metrics, it is not possible to supply beforehand a fully-equipped set of them (since metrics shall be introduced depending on each case) to provide a data source that helps to answer specifically the instantiated questions from an unbiased and quantitative point of view. As an example, let us consider a sample application to evidence its feasibility.

Let us suppose that we want to evaluate the quality of the TripAdvisor Android app in terms of its observance of user's privacy while retrieving his or her physical position. From the partial instantiation of the GQM, we obtain:

G.1: "Monitor the quality of the product for the purpose of assessing its compliance with the privacy policy from the point of view of the user in the context of the mobile execution environment".

Possible questions are:

Q.1: "What is the current performance of the application with respect to privacy compliance?"

Q.2: "Is the current performance of the privacy compliance satisfactory from the viewpoint of the user?" and

Q.3: "Is the current performance of the privacy compliance acceptable from the viewpoint of the application market?"

We can then define the following metrics:

M.1: Amount of time spent on accesses to GPS data while the application is working, and

M.2: Amount of time spent on network access while the application is working.

Once we complete the definition of the quality attributes and the corresponding association with quality metrics, we plan to conduct a piloted evaluation.

Our experiments include the implementation of Android applications as suitable test bed to define and measure product quality metrics. Then, this approach will be extended to evaluate a number of Open Source mobile applications taken from real mobile app stores, to evaluate them using the selected quality attributes and their corresponding quality metrics. The selected applications should represent a variety of categories to cover a wider range of user and developer profiles, and they must be distributed under Open Source licenses, so that access to the source code can be guaranteed to retrieve it and execute the analysis (Corral 2012). Finally, we will correlate the values of such metrics with the level of accomplishment of the apps in the market, which may be denoted by parameters like the number of downloads and customer ratings. This information may be extracted from the data exposed by application stores (Harman 2012). With this study, we will investigate whether apps with better quality metrics are those achieving higher success in the application stores, providing a partial verification of the applicability and usefulness of the Mobile App Quality Model.

In this chapter we suggested to leverage the quality strategy set by the ISO/IEC 25010 standard to sketch a relationship between the app store quality requirements with the standard's quality characteristics as an effective way to assure the fulfillment of the market requirements. Utilizing the Quality Function Deployment methodology, we associate the quality requirements from mobile app stores with the quality characteristics of the ISO/IEC 25010 standard, and we ranked the outcome of such association to identify what characteristics of the software product have the highest relevance in the scope of what is required by the mobile application stores. Finally, we utilized this association and ranking to introduce a mobile-specific, standard-based software quality model that aims to provide the means to appraise the quality of the mobile software product based upon real-world market requirements. The definition of

the Mobile App Quality Model may be useful to relate the quality expectations from the mobile application markets with measurable attributes, to assist developers in the production of mobile apps that suit the requirements imposed by different app stores.

Software developers require the means to measure the quality of the mobile software product from a point of view that considers the characteristics and particularities of the mobile environment and the mobile business. Identifying the most relevant software quality characteristics from the publishing policies of major mobile app stores, and organizing them in a Mobile App Quality Model aims to contribute on the enhancement of the development and assurance processes that guarantee the production and delivery of successful mobile-specific, market compliant mobile applications.

5.5 Chapter bibliography

(Akao 1994) Akao, Y.; (1994). *Development history of quality function deployment. The customer driven approach to quality planning and deployment.* Asian Productivity Organization. ISBN: 92-833-1121-3.

(Basili 1994) Basili, V.; Caldiera, G.; Rombach D.; (1994). The Goal Question Metric approach. In *Encyclopedia of Software Engineering*. pp. 528-532. Wiley. doi:10.1002/0471028959.sof142

(CIRI 2007) Creative Industries Research Institute (2007) *Quality function deployment.* Auckland University of Technology. Available online: http://www.ciri.org.nz/downloads/Quality%20Function%20Deployment.pdf (Accessed on August 20th, 2013).

(Corral 2012) Corral, L.; (2012) Using software quality standards to assure the quality of the mobile software product. In *Proceedings of the 3rd annual conference on Systems, programming, and applications: software for humanity (SPLASH '12).* pp. 37-40. ACM. doi:10.1145/2384716.2384734

(Corral 2013) Corral, L.; Sillitti, A.; & Succi, G.; (2013) Using a partially instantiated GQM to measure the quality of mobile apps. In *Proceedings of the 25th International Conference on Software Engineering and Knowledge Engineering (SEKE 2013).* pp. 520-524. Knowledge Systems Institute.

(Harman 2012) Harman, M.; Jia, Y.; & Zhang, Y.; (2012) App store mining and analysis: MSR for app stores. In *Proceedings of the 2012 9th IEEE Working Conference on Mining Software Repositories (MSR).* pp. 108-111. IEEE. doi:10.1109/MSR.2012.6224306

(ISO 2011) International Organization for Standardization (2011) *ISO/IEC 25010:2011. Systems and software engineering. Quality requirements and evaluation system and software quality models.* Available online: http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=35733 (Accessed on September 7th, 2013).

(Karlsson 1997) Karlsson, J.; (1997). Managing software requirements using quality function deployment. *Software Quality Journal* vol. 6 (4) pp. 311-326. Kluwer Academic Publishers. doi:10.1023/A:1018580522999

(Kivinen 2008) Kivinen, T.; (2008). *Applying QFD to improve the requirements and project management in small-scale project.* Master Thesis. Department of Computing Sciences, University of Tampere, Finland.

(Mandic 2010) Mandić, V.; Basili, V.; Harjumaa, L.; Oivo, M.; & Markkula, J; (2010) Utilizing GQM+ strategies for business value analysis: an approach for evaluating business goals. In *Proceedings of the 2010 ACM-IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM '10).* pp. 20. ACM. doi:10.1145/1852786.1852813

(MOETIJ 2011) Ministry of Economy, Trade and Industry, Japan. (2011). *Investigative report on measure for system/software product quality requirement definition and evaluation.* Technical Report. Software metrics advanced project. Product quality metrics working group.

(Myint 2003) Myint, S.; (2003). A framework of an intelligent quality function deployment (IQFD) for discrete assembly environment. *Computers & Industrial Engineering.* vol. 45 (2) pp. 269-283. Elsevier. doi:10.1016/S0360-8352(03)00035-4

(Ramires 2005) Ramires, J.; Antunes, P.: & Respício, A.; (2005) Software requirements negotiation using the software Quality Function Deployment. In *Proceedings of Groupware: Design, Implementation, and Use (CRIWG 2005).* Lecture Notes in Computer Science vol. 3706. pp. 308-324. Springer, Heidelberg. doi:10.1007/11560296_25

(Xiong 2008) Xiong, W., & Wang, X. T. (2008) Software requirements management using QFD: A process perspective. In *Proceedings of the International Symposium on Computational Intelligence and Design, 2008 (ISCID '08).* vol. 2. pp. 295-299. IEEE Press. doi:10.1109/ISCID.2008.123

Chapter 6: Measuring Product Quality Of Custom Applications

After the definition of the mobile-relevant quality characteristics, we need to conduct evidence-based research that unveils what is the performance standard of some of the identified parameters, measured on mobile applications. These insights may be gained by means of experiments and empirical studies that implement the partially instantiated GQM to identify mobile-relevant software metrics to be later collected using both custom or real world applications.

In this chapter we describe two experiments that implement the partially instantiated GQM approach in a setting that involves actual mobile products coded and customized in our research center. With the partially instantiated GQM we defined a set of metrics based on the ISO/IEC 25010 quality attributes previously analized in Chapter 5, guaranteeing the implementation of both the development quality and quality in use models. Additional surveys implementing the partially instantiated GQM on real applications are included in Chapter 7.

6.1 Case study 1: Measuring performance of a custom application

In the first case study, we concentrate on measuring performance, a parameter that belongs to the quality characteristic "Performance Efficiency" and sub-characteristic "Time Behavior" from the development-oriented quality model, and to the quality

characteristic "Efficiency" from the user-oriented quality model. As a case study, we measured the performance of web-based Android applications with respect of a native Android application (Corral 2011), (Corral 2012a).

The overall performance of an application is an important parameter that helps to describe how usable, efficient and satisfactory an application could be. Performance can be measured in a number of ways: execution time, memory usage or CPU load are parameters that typically yield useful values for performance assessment (Frisiani 2011). Our study focuses on application's execution time, as this parameter impacts directly the user's experience when he uses an application, or when such an application interacts with hardware and software resources in the device.

Evaluating the execution time is not a task that merely requires sampling the time used by a routine to run. The task of appraising two machines, languages or techniques shall bear in mind the creation of a proper environment to maximize fairness, and make use of appropriate procedures for data interpretation (Fleming 1986).

6.1.1 Implementation of the partially-instantiated GQM

In this experiment, we need to analyze how long the application is taking to execute a given routine. From the partial instantiation of the GQM, we obtain as Goal:

> *G.1:* "Analyze the web-based and native mobile software product for the purpose of evaluating it with respect to the *efficiency*, from the view point of developers, in the context of *execution environment*".

The questions are set to survey the characteristics of the application and to evaluate it with respect to the selected attribute (execution time):

> *Q.1:* "What is the current *performance* of the web-based and native application with respect to its execution time?"
>
> *Q.2:* "Is the current *execution time* satisfactory from the viewpoint of the developer?"

As metric, it was selected the *execution time* obtained in milliseconds and measured using software according an operational definition that should be described in detail.

M1: Execution time (mSec)

To implement the experiment, we selected PhoneGap[16] as development tool and Android OS as target platform, thanks to their openness, flexibility and availability. To complete our work, we executed two applications in an experimental environment, measuring and comparing the execution time as the selected performance parameter.

6.1.2 Experimental setting

As implementation of the case study, two Android applications were coded using JavaScript programming, and regular Java programming respectively, and they were deployed in a real mobile terminal. Each application is able to call subroutines that exercise specific features in the mobile device. Each time a routine is launched, the application records the time required to complete the job. To acquire this value, we instrumented the code by adding sentences to take a time sample immediately before calling the routine (t_1), and promptly after obtaining a successful answer of its completion (t_2).

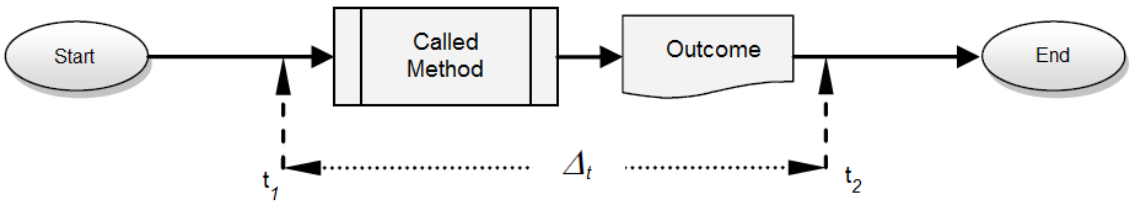


Figure 15: Operational definition of the measured job

Figure 15 represents the operational definition to determine the starting and the ending instants that bound the measured task. In this manner, we assure to take into consideration the complete amount of time that exercises each feature, from its original invocation to its successful completion, including a tangible response. The final execution time is calculated from the difference between the two time samples ($\grave{a}_t$).

[16] http://www.phonegap.com

6.1.3 Methodology

We based our analysis and data interpretation on the recommendations proposed by Fleming (1986) to compare performance between two machines according to a metric. First, it is necessary to normalize the results to a "known machine" to represent relative system performance; then, to calculate the geometric mean to average the normalized numbers; finally, to use this geometric mean to compare relative performance and draw conclusions. Since Java is the language natively supported by Android, Java was considered as known machine. Thus, normalized values are calculated taking the time samples achieved by Java and JavaScript, and dividing each one by the Java value.

6.1.4 Mobile application

The mobile application consists of a graphic user interface furnished with buttons through which an operator can launch a routine to access a given feature. Both mobile applications were implemented in such a way to provide the same user experience (Figure 16). The goal of each routine is to send a request to a hardware or software resource, and to retrieve a response or information as an acknowledgment that the resource was successfully accessed. The application reports and records the amount of time that was required to complete the job.

To accomplish a comprehensive analysis in the mobile terminal, we considered resources from different categories:

a) Hardware access: access to accelerometer, launch a sound notification, trigger vibrator.

b) Network access: request data from GPS, request network information, and

c) Data access: write data into a file, read data from a file, retrieve data from a content provider.

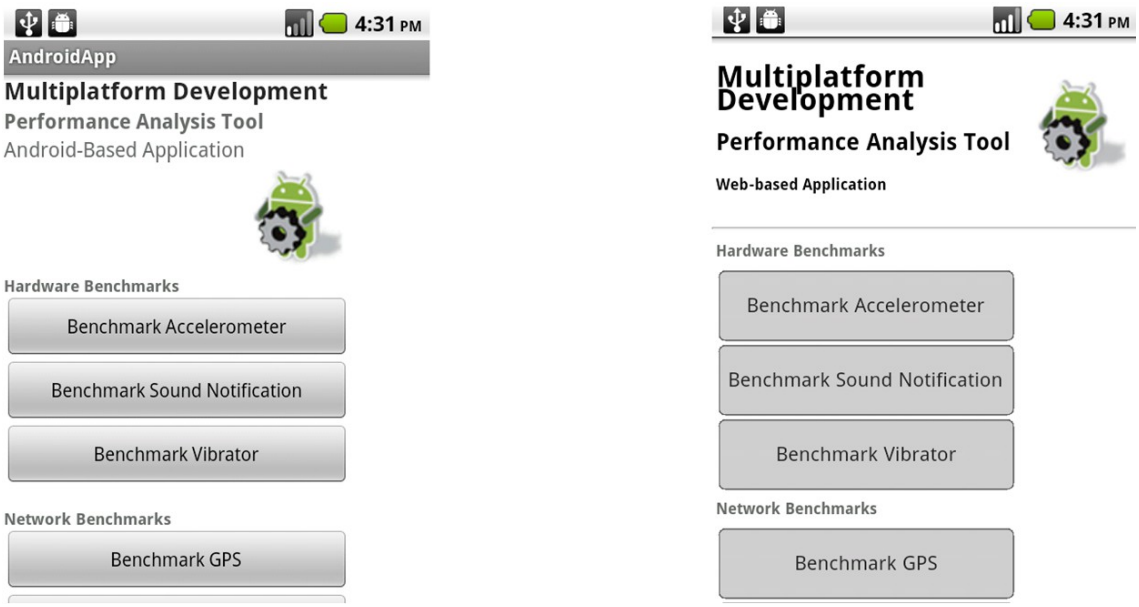


Figure 16: Android native application and PhoneGap web application

When a button is pressed, a method attempts to access the selected device or resource. As per our operational definition, the application samples the time immediately before calling each method and after its successful completion, storing the time sample in a file for future use. To obtain the time samples, we selected the method that returns the value of the most precise system timer available. Java is able to obtain time samples in a resolution of nanoseconds, but the highest resolution of the JavaScript timer is of milliseconds, so the last unit was chosen. Each routine was exercised 1000 times to achieve statistically significant results.

The selected test bed was a cellular phone HTC Nexus One, equipped with Android OS 2.2. To ensure repeatability and reproducibility of this experimentation, we also run our tests in a HTC Magic cellphone where we could observe consistent results, yet they were kept for reference only and are not reported in this paper.

6.1.5 Data analysis

Results are summarized in Table 11. We report arithmetic means in milliseconds and standard deviation to illustrate the distribution of the data. To conduct performance

analysis, we considered exclusively the data in relative time units obtained after normalizing all time samples with respect to the Java application, and their corresponding geometric means. As reference machine, the geometric mean of Java tasks must remain constant in 1. For each JavaScript job, the geometric mean will show a value less than 1 if it is statistically more efficient, or greater than 1 if it is statistically less efficient than the same task executed by the reference machine.

Table 11. Evaluation of execution time of Android native and web application

Measured Job	Arithmetic Mean (milliseconds)		Standard Deviation		Geometric Mean (relative)	
	Native App	Web App	Native App	Web App	Native App	Web App
Access to accelerometer	0.7136	2.0021	0.9984	3.0025	1.0000	2.5974
Launch sound notification	18.4835	26.7481	13.3665	47.5036	1.0000	0.6534
Trigger vibrator	1.5134	3.2222	1.2234	4.1248	1.0000	2.2593
Request data from GPS	2.1881	809.2352	6.7244	12.5523	1.0000	528.9298
Request network information	1.1015	1.01419	1.2052	0.6096	1.0000	1.1044
Write a file	4.7146	7.9221	9.2085	6.4558	1.0000	3.3657
Read a file	13.3036	255.7381	13.8829	74.1943	1.0000	29.9005
Retrieve data from contact list	95.8686	1841.4689	13.8747	491.5454	1.0000	18.0001

Values in Table 11 show that the web application only succeeded in executing equal or better than the native application in one routine (e. g. launching a sound notification, 35% faster). For the rest, there is a performance decay that goes from slight (e.g. requesting network information, 10% slower) to very significant (e.g. accessing the GPS sensor).

6.1.6 Discussion

To approximate the root cause of this difference, we analyzed the inner structure of the resource call at code level for each version, noticing that while Java programming typically uses native methods to directly access the specified resource, JavaScript is allowed to access resources only by following an execution path that implements at

least one callback. This introduces waiting and execution times for calling the method, going through the callback path and waiting for the result to reach the original requestor. Execution time becomes especially high when accessing a resource that involves a complex series of calls to invoke the associated API.

According to PhoneGap's architecture, a JavaScript method defined at user space is sent as parameter to a foreground executive method called PhoneGap.exec(); this executive method invokes a JavaScript function (i.e. prompt()), with a twofold objective: to generate an event that may be caught by the PhoneGap's native engine, and to send the necessary parameters via a JSON string. Then, PhoneGap's native layer implements the capture of the JavaScript method and its arguments, and delegates the call to the corresponding controller or API, as seen in Figure 17.

At the same time, a callback defined on the application's web view is notified when the JavaScript method is called, before the actual result (a prompt dialog) is shown. Finally, the native Java method checks the parameters sent, executes the request, and passes the return value back to original JavaScript caller using a string message.

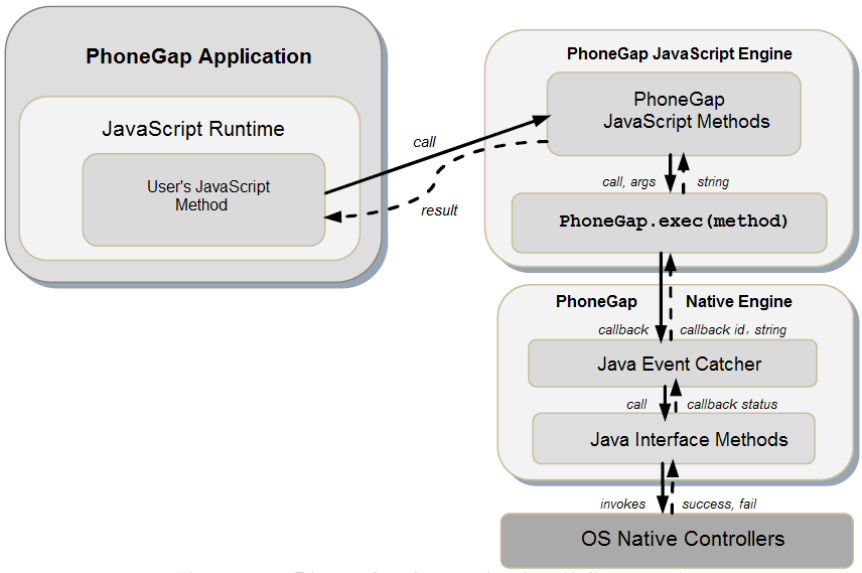


Figure 17: PhoneGap's method call flow path

Tracing the method through the callback tree and delivering the result, in combination with the resource's usual response time, introduces overhead that might make this architecture prone to become expensive, especially when resources need to access a

complex execution tree. Other web-based frameworks that rely on this structure to expose resource-specific methods to the web browser view should expect similar performance losses in their applications.

Although it is acknowledged that execution time increases on web applications (and in certain conditions critically), it is important to note that the experimental setup also shows that the performance penalty on a number of features commonly used is minor. These results agree with the discussion proposed in (Charland 2011), where it is said that web-based mobile applications are more suitable for business applications, or applications that do not make extensive use of resource-hungry code (for example, rendering 3D graphics, or performing other heavy hardware-consuming operations).

6.1.7 Case study conclusions

Mobile development based on web technology is a rich field for discussion, presenting both advantages and disadvantages. On one hand, it gives the opportunity of broadening the scope of a single application to a wider range of potential targets, overcoming the need of repeating platform-specific efforts through the software life cycle. On the other hand, current development tools still present limitations, particularly accessing device-specific features and interacting with other software resources. Moreover, mobile applications based on web technologies have been reported as showing important losses in performance, affecting the overall user experience.

In this case study, we analyzed specific performance matters on web-based mobile applications, showing the extent to which the execution time of a task coded using web-based programming increases with respect to an identical job developed using native, target-specific tools. In our experimentation, we exercised hardware and software features in a cellular phone, acquiring a dataset that allowed us to identify the cases in which execution time rises.

Using principles of machine benchmarking, we determined that in 7 out of 8 routines, web-based implementation was slower than the native one, observing that the execution time increases due to an architecture that requires to invoke methods using

at least one callback and waiting for its response. This overhead grows when resources need a complex execution tree, both to be accessed and to send an answer back to the requester. For general-purpose business applications, even though it is expected a performance penalty, it is noted that such penalty will be slight.

In the context of the partially instantiated GQM, through this experimentation we were able to analyze Android web-based and native applications for the purpose of evaluating their efficiency, answering questions about the performance in terms of execution time, positioning its results in the context of the development-based mobile software quality model. The results of this case study are published in detail as a full paper in conference proceedings (Corral 2012b).

6.2 Case study 2: Measuring energy consumption of a custom application

In the second case study, we focused on measuring energy consumption, a parameter that belongs to the quality characteristic "Performance Efficiency" and sub-characteristic "Resource Utilization" from the development-oriented quality model. However, in this implementation we will focus on the quality characteristics of "Efficiency" and "Freedom of Risk" and "Economic Risk Mitigation" from the user-oriented quality model.

This quality characteristic is of utmost important in the regular use of a mobile device: cellular phones are normally carried by their owners, most of the time away from a permanent power source like an AC power outlet. In this way, the normal use of these devices implies a constant energy demand that will invest the battery reserves until they finally die. Users would like a wise utilization of energy to allow their phones to be more and more autonomous. This may imply using expensive, state-of-the-art batteries that last longer, or configuring energy profiles that maximize the profit from a limited power source at the cost of having less features or lower performance. Optimizing power consumption has been investigated at different levels, e.g., system, circuits, processors, memories, displays, sensors and software (Olsen 2006), with the scope of optimizing the

consumption of energy resources to extend the battery life. Nonetheless, to be able to optimize the power consumption, we have first to measure it accurately. To this end, a core requirement is a good understanding of where and how the energy is used (Carroll 2010) (Zhang 2010) (Jung 2012) (Pathak 2012) (Yoon 2012) to know precisely how the resources are utilized and to discover improvement needs.

6.2.1 Implementation of the partially-instantiated GQM

In this experiment, we aim to analyze the battery drain of a mobile device in a certain time frame. Specifying this as part of the partial instantiation of the GQM, we define as Goal:

> *G.1:* "Analyze the mobile software product for the purpose of evaluating it with respect to the *energy consumption*, from the viewpoint of users, in the context of execution environment".

The questions are set to survey the characteristics of the device and to evaluate it with respect the outstanding attribute (battery drain):

> *Q.1:* "What is the current *energy consumption* of the device with respect to its battery drain?"
>
> *Q.2:* "Is the current performance of the *energy consumption* satisfactory from the viewpoint of the user?"

As metric, we selected the *battery charge units*, obtained in percentage in a scale from 0% to 100%. This value is obtained using the measurement instruments provided by the device software interfaces.

> *M1:* Battery Charge Units (%)

To furnish the necessary data to fulfill our GQM instance, we implemented our solution by means of a regular Android mobile application that analyzes the system and characterizes the discharge cycle. We developed CharM (Roeggla 2012), an Android app that records the battery level and additionally collects specific data from the OS to survey the components deem to have an impact on the discharge curve of the battery.

6.2.2 Data sources

The CharM application runs a simple user interface that triggers a service that records the following data in fixed intervals of one minute.

- *Timestamp:* A UNIX timestamp to associate the reading with a time.
- *Battery level:* The current charge level of the battery in percent relative to a full charge.
- *Processor load:* The current load of the processor in percent, i.e. the time the processor has actually been working versus the total time.
- *Processor frequency:* The current CPU frequency, as the system adjusts the frequency dynamically.
- *Memory usage:* A value which indicates what share of the total available RAM is in use.
- *Airplane mode:* A logical value which indicates if the phone is connected to a mobile network
- *Display active:* A value that shows whether the screen is currently on, as well as the brightness intensity.
- *WiFi enabled:* A logical value that indicates whether the WiFi service is enabled.
- *Bluetooth enabled:* A logical value that specifies whether the Bluetooth service is enabled.
- *GPS enabled:* A logical value which indicates if the GPS service is enabled.
- *Amount of bytes transmitted and received:* Indicates the amount of data in bytes that has been sent and received on the wireless interface.

6.2.3 Mobile application

To implement the CharM application, several design factors were considered. It was critical to guarantee that the application is run repeatedly in the time interval specified, without being terminated by the system or interfering with the user blocking the user

interface. We had to ensure that collecting the statuses will not block the user interface and will not be interrupted by the system. Finally, we had to establish the way in which the data should be collected. In the Android OS there are two main sources for important data for measuring energy consumption. First, the Android SDK exposes the current battery level or the status of various components like the screen or the different wireless interfaces, and delivers relevant data for the analysis of power consumption. The second source for information is the Linux kernel. As a UNIX-like OS, the kernel provides us with the virtual proc-filesystem which is mounted at the directory /proc directly under the root directory. This directory contains files with assorted information about the current status of the system. For instance, the file /proc/stat contains information about the CPU usage and the file /proc/meminfo about the allocation of main memory. We choose to survey the system at this level, allowing the OS to work as an intermediate level of abstraction that permits a less hardware-dependent analysis, more reliable on diverse product families (Figure 18).

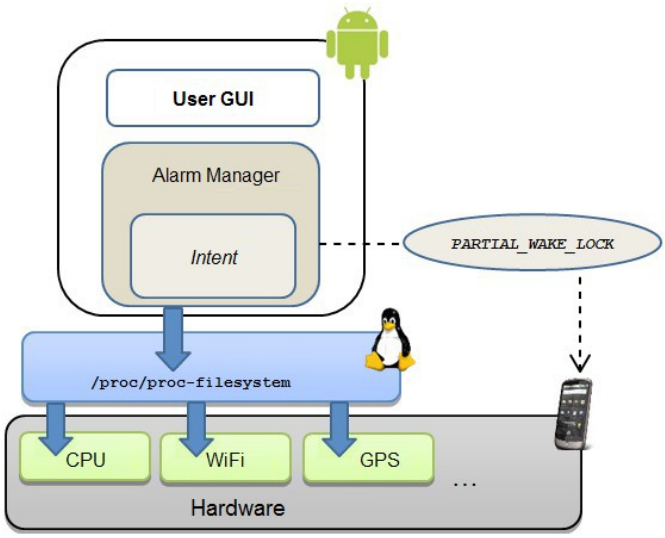


Figure 18: Android application architecture

To explain how these files can be utilized and how the measurements can be interpreted from them, we present a separate analysis for each one:

- *Processor usage information:* The data related to processor usage can be read from the file /proc/stat. This file contains one or more rows, depending on the

number of processors that the system has. The values in each row indicate the time each core has been working or idle since the launch of the system. To get a value for CPU usage, we calculate the ratio between the time that the processor has been active, and the time the processor has been idle.

- *Current processor frequency:* In a different way, the current processor frequency cannot be found in the directory /proc but rather in the file /sys/devices/system/cpu/cpu0/cpufreq/scaling_cpu_freq, which contains a value for the current frequency in kilohertz (kHz).
- *Memory allocation data:* The data for memory allocation information can be read from /proc/meminfo. This file contains several lines, where each line contains a name and a value in kilobytes. The lines in which we are interested are named MemTotal and MemFree. From their values, we can easily calculate the percentage of total memory which is currently in use.
- *Data on wireless interface use:* The file /proc/net/dev contains a row for each network interface. It contains the number of bytes that have been sent and received by every interface since the system was started.
- *OLED screen and backlight status:* To survey the screen, we can acquire the on/off status out from the PowerManagement class, while the value set for the backlight brightness can be taken from an integer value stored in the file /sys/devices/virtual/leds/lcd-backlight/brightness.

Once the data sources have been identified, it is necessary to define how the collected data should be organized and aggregated for further analysis. We implemented a process in which the data can be surveyed and stored minimizing the energy consumption associated to the data collection effort. With this implementation, the measurements are written and appended to a file located on the removable SD card using the CSV format, using a tabular, plain-text way. While this technique is the most suitable for our needs, it has the disadvantage that it does not allow real time analysis: the data need to be collected first, stored in the SD card, and transferred to a computer at the end of the experiment in order to execute further analysis.

6.2.4 Experimental setting

CharM was deployed in three Android-operated devices to collect measurements on different product families with diverse versions of this OS:

- *HTC Nexus One* cell phone, with Android 2.3.7, powered by a Li-Ion 1400 milliampere-hour (mAh) battery.
- *Samsung Galaxy* cell phone, with Android 4.0.4, powered by a Li-Ion 1750 mAh battery.
- *Nexus 7* tablet, operated by Android 4.2.1, powered by a Li-Ion 4325 mAh battery.

There has been extensive research that shows that data collection and visualization is of critical importance when conducting an empirical study (Scotto 2006) (Sillitti 2011). We established a solid data collection effort divided in three stages:

1. In the first stage of our experimentation, we installed CharM in the listed devices. We asked the users to recharge them to the full battery capacity, and to utilize the cellular phone normally in the time frame of 2 hours. CharM surveyed the status of the system every minute. This survey permitted to create an operational baseline that illustrates the relationship of the discharge cycle of the cellular phone with the system's resources utilized in normal, day-to-day conditions: occasional use of cellular radio and 3G connection, intermittent WiFi usage, and regular use of some of the mobile applications.

2. In the second stage, users were asked to recharge their devices to the full battery capacity and to utilize them in Airplane Mode (i.e., disabling all communication interfaces) in the time frame of 2 hours. CharM surveyed the status of the system every minute. Through this practice, we created a baseline for the discharge cycle of the cellular phone resources in strict economy mode.

3. Finally, in a third stage, the devices were as well recharged to the full battery capacity, and the status of each phone was surveyed by CharM every minute for 2 hours. In this stage, we exercised a series of software routines that specifically

stress a component to observe the relation between the system's energy consumption with the power toll of such element in conditions of high demand (Table 12).

Table 12. Energy consumption stressing routines

Component	Routine
CPU	High-load, maximum frequency CPU stress
OLED screen	Maximum screen brightness
Multimedia playback	Continuous video playing
WiFi interface	Systematic WiFi transmission
GPS interface	Systematic GPS polling

To isolate effectively the power demand of each surveyed element, when a component was stressed we disabled all others (e.g., deactivating the WiFi or GPS interfaces, or turning off the screen). When a component could not be disabled (e.g., CPU), we minimized its utilization (e.g., by executing only the stressing routine and CharM under minimum CPU frequency). To ensure repeatability of our experimentation, we run every test ten times on each cellphone observing consistent results. Reproducibility is guaranteed since the three test beds belong to different product families. In total, we achieved more than 500 hours of tests, with no major discrepancies observed among similar scenarios.

6.2.5 Data analysis

To simplify the data analysis and interpretation, we illustrate the most important results in Table 13. This table shows a summary that includes the results obtained from executing the test scenarios in the three devices. It is important to note that these percentages do not represent absolute numbers, but a proportional value with respect to the general battery discharge cycle. The capacity of the battery and the power demand of the features may inject variations among product families. However, all results are consistent, although we can appreciate that as the device's model evolves,

enhancements on power management policies, hardware components and the capacity of the battery delivered noticeable savings on the overall system's power consumption.

Table 13. Percentage of battery discharge in two hours

Execution Mode	Nexus One	Nexus 7	Galaxy Nexus
Normal mode	10%	2%	4%
Airplane mode	2%	0.2%	1%
CPU stress	40%	17%	26%
OLED stress	35%	24%	20%
Video playback stress	12%	7%	10%
WiFi stress	24%	22%	32%
GPS stress	17%	10%	15%

With these results at hand, we can calculate an average of the readings to sort the components by energy consumption. Table 14 shows a ranking of the surveyed elements, sorted in descending order, from the hungriest to the least power demanding features. From it, we can identify that three components were noted to particularly drive the battery discharge cycle. They are the CPU, the OLED display and the WiFi interface. CharM provided data that ensure a clear view of the operational status of the system, permitting to isolate unambiguously that such components contributed the most to the system's energy toll.

Table 14. Ranking of energy hungry components

Component	Impact to discharge cycle
1. CPU stress	27.66%
2. OLED stress	26.33%
3. WiFi interface	26.00%
4. GPS interface	14.00%
5. Video playback	9.60%
6. Normal mode	5.33%
7. Airplane mode	1.06%

6.2.6 *Case study conclusions*

The information delivered by CharM opens the doors for further analysis on the energy utilization quota for each surveyed feature, useful to design economy profiles that can be manually or automatically set depending on how much energy the user is willing to spend. A primary opportunity to enhance CharM is to add features that may improve the survey of the system, including more parameters that can be taken at OS level, for example the temperature of the battery or the status of all the leds in the device. Moreover, we also foresee to improve the user experience by providing enhanced reporting features like plots, tables and customized reports.

In this case study, we illustrated a technique that builds upon direct measurement and energy models to furnish a software-driven approach to measure the impact of each relevant working components in the overall power consumption of the unit. This solution proposes an original approach since it places the survey at user space, but collecting data from sources at OS level. This facilitates a less hardware-dependent analysis, eases its maintenance, and makes it reliable on diverse product families of phones and tablets operated by Android, yet it is easily expandable to other environments operated on top of the Linux kernel.

Measuring the battery level supplies little understanding of the energy consumption of a device. CharM contributes to move towards a better comprehension of how the energy is used in a mobile device, delivering a relationship between the overall system consumption toll and the active components that mainly influence it. With this work, we deliver a characterization tool that eases the identification of components particularly energy consuming, helping to design execution profiles, collect relevant data for green metrics, and identify eventual optimization needs.

In the context of the partially instantiated GQM, through this case study we analyzed the battery drain of different Android devices, solving questions about their energy consumption, positioning the results in the context of the user-oriented mobile software quality model, providing data that helps to accomplish the goal of "analyzing the mobile

software product for the purpose of evaluating it with respect to the *energy consumption*, from the view point of users, in the context of execution environment".

The results of this case study are published extensively as a full paper in refereed workshop proceedings (Corral 2013). In addition, another study (Corral 2012c) permitted us to understand the importance of the energy consumption of mobile applications in a real-world, productive setting: we deployed a measurement mobile app that used Bluetooth for device discovery. Although the application performed correctly and it helped users to accomplish their goals, serious concerns about energy consumption declined deeply the satisfaction of the users of two independent teams, and after three months of rollout the app was not used due to battery draining issues.

Finally, we executed and experiment that permits to outline a relationship between execution time, analyzed in the first case study, and energy consumption, analyzed in the second case study. We implemented software benchmarks in Java and C and we exercised them in different execution scopes of the Android OS runtime. We measured the amount of energy required to complete each job to determine the energy consumed by each routine, and to know in what cases it is advisable to reallocate the processing job from a regular application to an external execution environment with the objective of saving energy and preserving the battery level (Corral 2014)

6.3 Summary and conclusions

The two case studies only represent a limited analysis of the quality attributes defined in Chapter 5; however, they provide interesting insights about how we can use the proposed partial GQM approach to set goals and questions and define metrics that can provide the necessary data for mobile-specific software quality analysis. Moreover, our case studies show how the collected metrics permitted to shed light and increase our understanding of two aspects of mobile application quality, namely performance and energy consumption, and how development practices, even though not considered in the Mobile App Quality Model, might contribute to create applications that demonstrate a better performance in relevant quality indicators.

6.4 Chapter bibliography

(Carroll 2010) Carroll, A.; & Heiser, G.; (2010) An analysis of power consumption in a smartphone. In *Proceedings of the 2010 USENIX conference on USENIX annual technical conference (USENIXATC'10).* pp 21. USENIX Association.

(Charland 2011) Charland, A.; & Leroux, B.; (2011) Mobile application development: web vs. native. *Communications of the ACM.* vol. 54 (5) pp. 49-53. ACM. doi:10.1145/1941487.1941504

(Corral 2011) Corral, L.; Sillitti, A.; Succi, G.; Garibbo, A.; & Ramella, P; (2011) Evolution of mobile software development from platform-specific to web-based multiplatform paradigm. *In Proceedings of the 10th SIGPLAN symposium on new ideas, new paradigms, and reflections on programming and software (ONWARD '11).* pp. 181-183. ACM. doi:10.1145/2048237.2157457

(Corral 2012a) Corral, L.; Janes, A.; & Remencius, T; (2012) Potential advantages and disadvantages of multiplatform development frameworks – A vision on mobile environments. In *Proceedings of the 3rd International Workshop on service discovery and composition in ubiquitous and pervasive environments (SUPE 2012), in connection with MobiWIS 2012.* Procedia Computer Science, vol 10. pp. 1202-1207. Elsevier. doi:10.1016/j.procs.2012.06.173

(Corral 2012b) Corral, L.; Sillitti, A.; & Succi, G.; (2012) Mobile multiplatform development: An experiment for performance analysis. In *Proceedings of the 9th International Conference on Mobile Web Information Systems (MobiWIS 2012).* Procedia Computer Science. Vol. 10. pp. 736-743. Elsevier. doi:10.1016/j.procs.2012.06.094

(Corral 2012c) Corral, L.; Sillitti, A.; Succi, G.; Strumpflohner, J.; & Vlasenko, J; (2012) DroidSense: a mobile tool to analyze software development processes by measuring team proximity. In *Proceedings of the 50th international conference on Objects, Models, Components, Patterns (TOOLS 2012).* Lecture Notes in Computer Science, vol. 7304. pp. 17-33. Springer-Verlag Berlin / Heidelberg. doi:10.1007/978-3-642-30561-0_3

(Corral 2013) Corral, L.; Georgiev A.B.; Sillitti, A.; & Succi, G.; (2013) Method reallocation to reduce energy consumption: An Implementation in Android OS. In *Proceedings of the 2nd International Workshop on Green and Sustainable Software (GREENS 2013), in connection with ICSE 2013.* pp. 38-45. IEEE. doi: 10.1109/GREENS.2013.6606420

(Corral 2014) Corral, L.; Georgiev A.B.; Sillitti, A.; & Succi, G.; (2014) A method for characterizing energy consumption in Android smartphones and tablets. Accepted for publication in the *29th ACM Symposium on Applied Computing (SAC 2014).* ACM.

(Fleming 1986) Fleming P.; & Wallace J.; (1986). How not to lie with statistics: The correct way to summarize benchmark results. *Communications of the ACM.* vol. 29 (3) pp. 218-221. ACM. doi:10.1145/5666.567

(Frisiani 2011) Frisiani A. (2012) *Evaluation of computers (In Italian: La valutazione dei calcolatori).* University of Genoa. Available online: http://www.laser.dist.unige.it/Repository/IPI-1011/Frisiani-Appunti.pdf (Accessed on March 15th, 2011).

(Jung 2012) Jung, W., Kang, C., Yoon, C., Kim, D., & Cha, H.; (2012) DevScope: a nonintrusive and online power analysis tool for smartphone hardware components. In *Proceedings of the eighth international conference on hardware/software codesign and system synthesis (CODES+ISSS '12).* pp. 353-362. ACM. doi:10.1145/2380445.2380502

(Olsen 2006) Olsen, C.M.; & Narayanaswami. C.; (2006) PowerNap: An Efficient Power Management Scheme for Mobile Devices. *IEEE Transactions on Mobile Computing* vol 5 (7) IEEE. doi:10.1109/TMC.2006

(Pathak 2012) Pathak, A.; Hu, Y. C.; & Zhang, M. (2012) Where is the energy spent inside my app?: fine grained energy accounting on smartphones with Eprof. In *Proceedings of the 7th ACM european conference on Computer Systems (EuroSys '12).* pp. 29-42. ACM. doi:10.1145/2168836.2168841

(Roeggla 2012) Roeggla, T.; (2012). *Methods for the measurement of energy consumption and its optimization on Android smartphones.* Dissertation. Faculty of Computer Science, Free University of Bozen-Bolzano, Italy.

(Scotto 2006) Scotto, M.; Sillitti, A.; Succi, G.; & Vernazza, T.; (2006). A non-invasive approach to product metrics collection. *Journal of System Architectures.* vol. 52 (11) pp. 668-675. Elsevier. doi:10.1016/j.sysarc.2006.06.010

(Sillitti 2011) Sillitti, A.; Succi, G.; & Vlasenko, J.; (2011) Toward a better understanding of tool usage. In *Proceedings of the 33rd International Conference on Software Engineering (ICSE 2011).* pp. 832-835. ACM. doi:10.1145/1985793.1985917

(Yoon 2012) Yoon, C., Kim, D., Jung, W., Kang, C., & Cha, H. (2012) AppScope: application energy metering framework for android smartphones using kernel activity monitoring. In *Proceedings of the 2012 USENIX conference on Annual Technical Conference (USENIX ATC'12)*. pp. 36. USENIX Association.

(Zhang 2012) Zhang, L., Tiwana, B., Qian, Z., Wang, Z., Dick, R. P., Mao, Z. M., & Yang, L.; (2010) Accurate online power estimation and automatic battery behavior based power model generation for smartphones. In *Proceedings of the eighth IEEE/ACM/IFIP international conference on hardware/software codesign and system synthesis (CODES/ISSS '10)*. pp. 105-114. ACM. doi:10.1145/1878961.1878982

Chapter 7:

Measuring Product Quality in Real Market Applications

To complement the analysis performed in Chapter 6, we pursue to implement a survey of additional quality parameters in real world mobile applications. We propose a survey that consists in retrieving a sample of assorted mobile applications from Google Play (Android OS market) and calculating from them the mobile software quality metrics defined by a partially instantiated GQM. The selected applications should span in a range of purposes to cover a wider spectrum of user and developer profiles, and they must be distributed under Open Source licenses so that the access to the repositories can be guaranteed to retrieve the code and perform automated product analysis.

To guarantee an objective, artifact-driven approach, the measurement and analysis of the product using software metrics has been extensively discussed, providing a quantitative guide for analyzing the quality of both processes and products. Mobile-specific software metrics have been introduced too (cf. Chapter 3), but there is little evidence about their applicability and contribution as an efficient way to forecast the potential level of accomplishment of the software product in a real application market.

In this Chapter, we investigated the contribution of the code quality to the market success of mobile apps in the context the official Android application store. We analyze whether products that are better developed potentially have a better chance to succeed

in the market. To verify this claim, we retrieved the source code of 100 Open Source mobile applications, calculated a set of product metrics and obtained several market indicators to find a potential correlation. We investigated if there is a relationship between the product quality, denoted by a collection of code metrics, and the success of the product in the market, denoted by the number of downloads and user's ratings.

7.1 Product evaluation and market success

There are many ways to evaluate the potential success of the software product. Most of these approaches concentrate on evaluating different attributes or traits in the product to create a correlation with the impact of the product in its deployment environment (for instance, an application market). We can distinguish two families: commercial products and research works.

Several companies offer products that aim to evaluate the quality of mobile applications based on indicators of market success, user satisfaction and other business analytics. In this category, we can find solutions like AppAnnie[17] and Applause[18], that concentrate on assessing app store ratings and user's reviews to get a sense of customer's opinion and market impact. In the same field, AppBrain[19] claims to have a "low quality app detection filter" that detects automatically which apps are unlikely to be useful, but they do not make public what are the parameters upon which such a filter works.

Additionally, a number of research papers have investigated the relationship between product quality and product success utilizing code metrics. These research works have been conducted using different approaches and originated different results: Stamelos et al. (2002) executed an experiment to determine the relationship between the product size metrics and user satisfaction, finding a positive relationship; Nagappan et al. (2005) utilized static source code metrics to provide developers with indications for added confidence and higher product quality, showing that source code metrics provide feedback on important attributes for better test quality; Midha (2008) analyzed a set of

[17] http://www.appannie.com

[18] https://my.applause.com

[19] http://www.appbrain.com

Open Source projects and identified a negative correlation between the code complexity metrics and the ability of these projects to attract more developers; Barbagallo et al. (2008) also analyzed a set of Open Source projects to outline the relationship between software design quality, expressed by object oriented metrics, and project success, expressed by the number of downloads, page views and development activities, and found a negative correlation; Meirelles et al. (2010), analyzed the source code of a large set of Open Source projects to study the relationship between source code metrics (mostly object oriented metrics) and the attractiveness of the project (number of contributors and number of downloads): their results indicated a positive correlation between the values of the quality metrics and the attractiveness of the project, and in general, it was confirmed an important contribution of source code quality in the overall success of the project (Santos et al. 2012).

Even though some of the conclusions are different, as a body of research they all concur in the fact that measuring the product's source code delivers valuable insights to outline the relationship between the product quality and the success (i.e., impact, attractiveness) of a project in a productive setting. We want to build on top of this approach and translate it to the context of mobile applications and mobile app stores.

7.2 Relating mobile code quality and market success

We acknowledge that the success of a mobile software product in the application store relies on a set of complex and diverse factors that go beyond the product itself (e.g., originality, marketing strategy, visual identity, etc.). Most of the commercial, marketing and non-scientific literature available focuses on these concepts and pays little attention to the contribution of well-coded software in the success of the final product. We want to shed light on matter, investigating the impact of the quality of the source code in the success of the mobile software product.

7.2.1 Implementation of the partially-instantiated GQM

We pursue to know whether the mobile software product can be analyzed to relate measurable parameters of the software product with user expectations and application market compliance criteria:

> *G.1:* "Analyze the mobile software product for the purpose of evaluating it with respect to the product quality, from the view point of developers and customers, in the context of application market success".

Then we will set up a series of questions that help to analyze the goal in a finer level of granularity in two dimensions: product quality and application market success:

> *Q.1:* "What is the current performance of the application with respect to product quality?"
>
> *Q.2:* "What is the current performance of the application in the application store?"
>
> *Q.3:* "Does the source product quality contribute to the success of the mobile software product in the context of application markets?"

To define the metrics we need to set the criteria for establishing the values of product quality and market success. We assume that the product quality may be inferred by inspecting the application code and calculating the associated metrics, while the market success may be expressed by the analysis of the information provided by the application store, such as number of downloads, user ratings and other indicators.

To have an objective and quantitative notion of the quality of the code, we consider the values of well-grounded product metrics. Since mobile applications are coded utilizing the Object Oriented Programming paradigm (e.g., using Java, Objective C, Java ME, C♯, etc.), we propose to use the Chidamber and Kemerer (1994) set of metrics, one of the most referenced for this paradigm. The "CK" metrics suite is a widely accepted and validated standard for measuring object-oriented software systems. Several studies have been conducted to validate CK metrics both theoretically and empirically.

Moreover, this set of metrics is simple to use, and it has shown its usefulness in constructing prediction systems, for example, for size and number of defects (Succi 2005) (Scotto 2006).

The CK metrics suite comprises six metrics, which capture different aspects of object oriented systems, including complexity, coupling and cohesion.

- *M1: Weighted Methods per Class (WMC):* Complexity metric that is defined by the sum of the complexities of all class methods. The number of methods and their complexity is a predictor of how much time and effort is required to develop and maintain the class.
- *M2: Depth of Inheritance Tree (DIT):* Complexity metric that is defined as the maximum length from the node to the root of the tree. The deeper a class is in the hierarchy, the greater the number of methods it is likely to inherit, making it more complex to predict its behavior. Also, the deeper a particular class is in the hierarchy, the greater the potential reuse of inherited methods.
- *M3: Number of Children (NOC):* Complexity metric that is defined as the number of immediate subclasses subordinated to a class in the class hierarchy. The greater the number of children, the greater the reuse since inheritance is a form of reuse. The number of children depicts the potential influence that a class has on the design.
- *M4: Response for a Class (RFC):* Coupling metric defined as the number of methods that can be potentially executed in a response to a message received by an object of that class. If a large number of methods can be invoked in a response to a message, the testing and debugging of the class becomes more complicated, since it requires a greater level of understanding.
- *M5: Coupling between Objects (CBO):* Coupling metric defined for a class as a count of the number of other classes to which it is coupled. A class is said to be coupled with another class if either one accessed the others methods, or variables. Excessive coupling between object classes is detrimental to modular design and prevents reuse. The more independent a class is, the easier it is to reuse it in another application.

- *M6: Lack of Cohesion in Methods (LCOM):* Cohesion metric that measures the amount of method pairs which don't access the same instance variable minus the amount of method pairs which do access the same instance variable. Cohesiveness of methods within a class is desired, since it promotes encapsulation. Lack of cohesion implies that a class should probably be split into two or more classes. High cohesion usually relates with low coupling and vice versa.

To complement the CK suite, we incorporate the insights provided by other indicators, namely cyclomatic complexity, and size, denoted by the logical lines of code.

- *M7: Cyclomatic Complexity (CC):* Complexity metric that measures the number of linearly independent paths through a program's source code (McCabe 1976). This metric complements the complexity metrics from the CK suite, giving input values for WMC and providing a procedural point of view.
- *M8: Lines of Code (LOC):* Evaluates the size of a software product. It is given by the number of executable lines of code, excluding comments and blank lines.

The metrics described above can be organized in a "Code Quality Model". This model structures the metrics in four characteristics (complexity, coupling, cohesion and size) that will be used to appraise the product based on the quality of the source code of the application (Figure 19).

To analyze the success of a product in the market, we need to consider the different sources of information that permit us to infer if an application is "successful" in the market (Cambria 2013). A successful product is one that is deeply known, extensively used, and that allows the users to achieve their goals in the specific context of use.

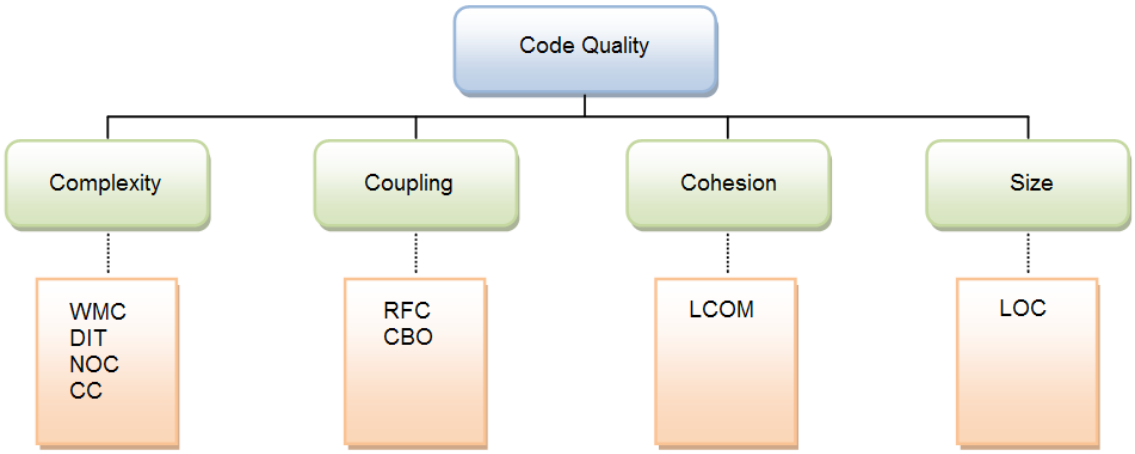


Figure 19: Product quality model based on source code metrics

To determine the level of accomplishment of a product, we rely on the information that is provided directly by the mobile storefront; as consequence its nature and availability are subject to changes depending on the analyzed application store. Typically, the following metrics can be obtained from the major mobile app stores:

- *M9: Number of Downloads (NOD):* A number that expresses how many times the application has been downloaded. Application stores rarely provide the exact number of downloads; however, some stores provide a rough approximation indicating the range of downloads that the app has reached. We take the middle value of such a range as an approximation for the penetration of the product, since it indicates the average number of people who have obtained and utilized the application.
- *M10 Number of Reviewers (NOR):* It is the number of users who have rated the product, this is, users who have explicitly fed a number to evaluate the application (see Application Rating). The number of reviewers is an indication of the penetration of the product, as it sketches the number of persons who have demonstrated interest on rating the application.
- *M11. Application Rating (AR):* It is the average rating obtained from users. Customers may rate an application by assigning a number from 0 (less satisfied) to 5 (very satisfied). The application rating is the sum of all ratings divided by the number of reviewers, and serves as an indicator of how satisfied is the user with the application.

These values are the numbers directly provided by the app store, and can be retrieved freely by querying the application name in the app store user interface. With them, we describe the "Market Success Model". The model structures the metrics on two characteristics (penetration and satisfaction) that will be used to assess the impact achieved by an application in the mobile market (Figure 20).

Figure 20: Selected market success model based on app store metrics

Providing relationship between the two models will permit us to understand if "better-coded" apps are likely to be more successful in the app market. To acquire the data that is necessary to answer this question, we need to carry out a survey that involves retrieving the source code of a set of mobile applications present in a large mobile application store, and calculating several source code metrics. Then, we need to analyze the market metrics given by the app store, and find an eventual relationship between the two sets of metrics.

7.2.2 Selected application market

From the different mobile platforms available in the market, for our study we selected the development and marketing environment of the Android OS. This selection relies upon the following reasons:

i. Android OS is an Open Source platform. This facilitates the possibility of finding a larger number of Open Source apps targeted to this operating system.

ii. Google Play, the official app market of the Android platform, offers assorted market metrics that are a fundamental input for our analysis. Such metrics can be retrieved freely by querying the application name in the Google Play website[20].

iii. Android OS permits to deploy applications coming from the official storefront but also from other repositories, which is essential to be able to retrieve the source code to measure and analyze the applications.

iv. Android OS has been deeply adopted by users, developers and mobile manufacturers, making this operating platform one of the most popular worldwide. This brings the opportunity of exercising our analysis in a wider variety of applications, and obtaining market metrics from a larger number of users.

v. Android OS development is mainly based on the Java programming, which allows the implementation of comprehensive source code metrics that permit a high understanding and better analysis of the quality of the code.

In summary, Android OS as target platform will provide us with the necessary openness, flexibility, availability, variety and market awareness to conduct our study.

With this, we established that two models in which the product quality shall be inferred by inspecting the application code and calculating product metrics, while the market success shall be expressed by the analysis of the indicators provided by the Google Play store. It is important to underline the fact that the origin of the data of the code quality model (application code) is totally unrelated to the origin of the data of the market success model (app store), ensuring independence between the two data sources.

7.2.3 Study of code quality and market success on open source Android applications

We conducted a survey on Android applications from the Google Play store, obtaining the selected source code quality metrics and relating them to the available market success parameters. The selected applications represented a variety of categories, always distributed under Open Source licenses so that access to the source code was

[20] `https://play.google.com/store/apps`

guaranteed. We correlated the values of such metrics with the level of success derived from the corresponding indicators obtained from the app store.

As of the second quarter of 2013, Google Play hosted nearly 1 million applications that span to a great variety of purposes and target audiences. Obtaining a sub-set of applications from this store is not a hard task, and thanks to the search feature offered by the store interface one can look for keywords like "Open Source", "FLOSS" or "FOSS" to retrieve apps that match the Open Source pre-requisite. However, even though an application matches the FLOSS criterion, Google Play does not necessarily provide a link to the code repository. To overcome this issue, we utilized an app repository that explicitly focuses on hosting Open Source applications that provides the necessary links to locate and retrieve the source code.

The F-Droid Repository[21] is a catalog of Free/Open Source software applications for the Android platform. The store contains links to both installation package and source code repository. The installation packages are verified and guaranteed to correspond to the source for the version available on the store's web site. Since F-Droid does not track users and devices, it is not possible to obtain any market metrics out of this store. In consequence, once we retrieved the source code of each application, we should look for it in the Google Play app store and obtain the market metrics from this application store.

To calculate the code metrics we used PROM. PROM is a system for automated data acquisition and analysis that collects both code and process measures (Sillitti 2003). For the purpose of our survey, we will focus on the functionality that analyzes the source code and computes the metrics (Scotto 2006). PROM calculates source code metrics based on three main components: a parser that extracts a high-level representation of the code; a database that stores the relations that represents the source code; and an analyzer, which calculates the metrics querying the database relations. PROM supports different programming languages, like C, C++, C♯, and Java (Scotto 2004), making it suitable for analyzing Android applications..

[21] https://f-droid.org

7.2.4 Data collection plan

The data collection for our survey was carried out in six steps according to the following operational definition (Figure 21)

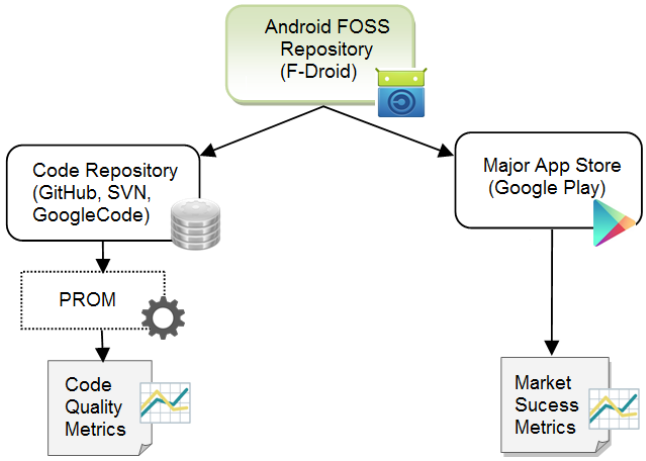


Figure 21: High-level data collection plan for source code metrics and app store metrics

1. Select a sample of 100 assorted Android applications from the F-Droid repository. The selected applications should represent a variety of categories to cover a wider range of profiles.

2. Determine if each one of the selected applications is also present in Google Play, matching the app version present in F-Droid. If a certain application is not showcased in Google Play, it should be dismissed and replaced by another application.

3. In F-Droid, locate the software repository (e.g., GitHub, SVN, etc.) where the source code of the application resides. Go to the repository and retrieve the source code, and verify if the Android application is mainly written in the Java programming language. If a chosen application is not coded in Java (e.g., mainly in C/C++ native code), it should be dismissed and another one should be selected to replace it.

4. Input the source code of the project into the PROM framework. Execute the source code calculation routine. The calculated source code metrics (WMC, DIT, NOC, RFC, CBO, LCOM, LOC, and CC) should be stored for further reference.

5. Consult individually the Google Play home page for each selected application. Inspect and retrieve the indicated values of the app market metrics available in the store (NOD, NOR, and AR).

7.2.5 Data analysis

To improve the organization of this section, the full raw data set of the metrics obtained from the PROM system and the Google Play Market is included in Appendix A. The data calculation and analysis in this section was done utilizing the R statistics package.

a) Descriptive analysis of product metrics

The first question of our GQM model pursues to understand what is the current performance of the mobile applications with respect to product quality. In consequence, we defined a quality model based on software metrics calculated directly from the software product.

Descriptive statistics give us the necessary values to report the performance achieved by our set of applications with respect to our quality model. We display a summary of the most relevant statistics (Table 15) and the non-parametric data distribution (Figure 22). In this way, we have enough information to give answer to question Q.1.

Table 15. Descriptive statistics for product oriented metrics of 100 Google Play applications

	Complexity				Coupling		Cohesion	Size
	WMC	DIT	NOC	CC	RFC	CBO	LCOM	LOC
Minimum	5.00	1.30	0.00	1.26	9.33	4.54	2.41	381
1st Quartile	10.34	1.92	0.15	1.84	17.04	8.63	15.71	4426
Median	13.40	2.19	0.32	2.02	20.04	10.52	28.17	11526
3rd Quartile	17.76	2.68	0.48	2.40	23.10	12.47	55.55	21928
Maximum	36.33	4.24	1.85	3.46	36.67	18.00	160.33	584144
Mean	14.35	2.30	0.33	2.13	20.52	10.63	39.67	22013
Std. Dev.	5.75	0.58	0.26	0.46	5.20	2.83	33.23	57079.69

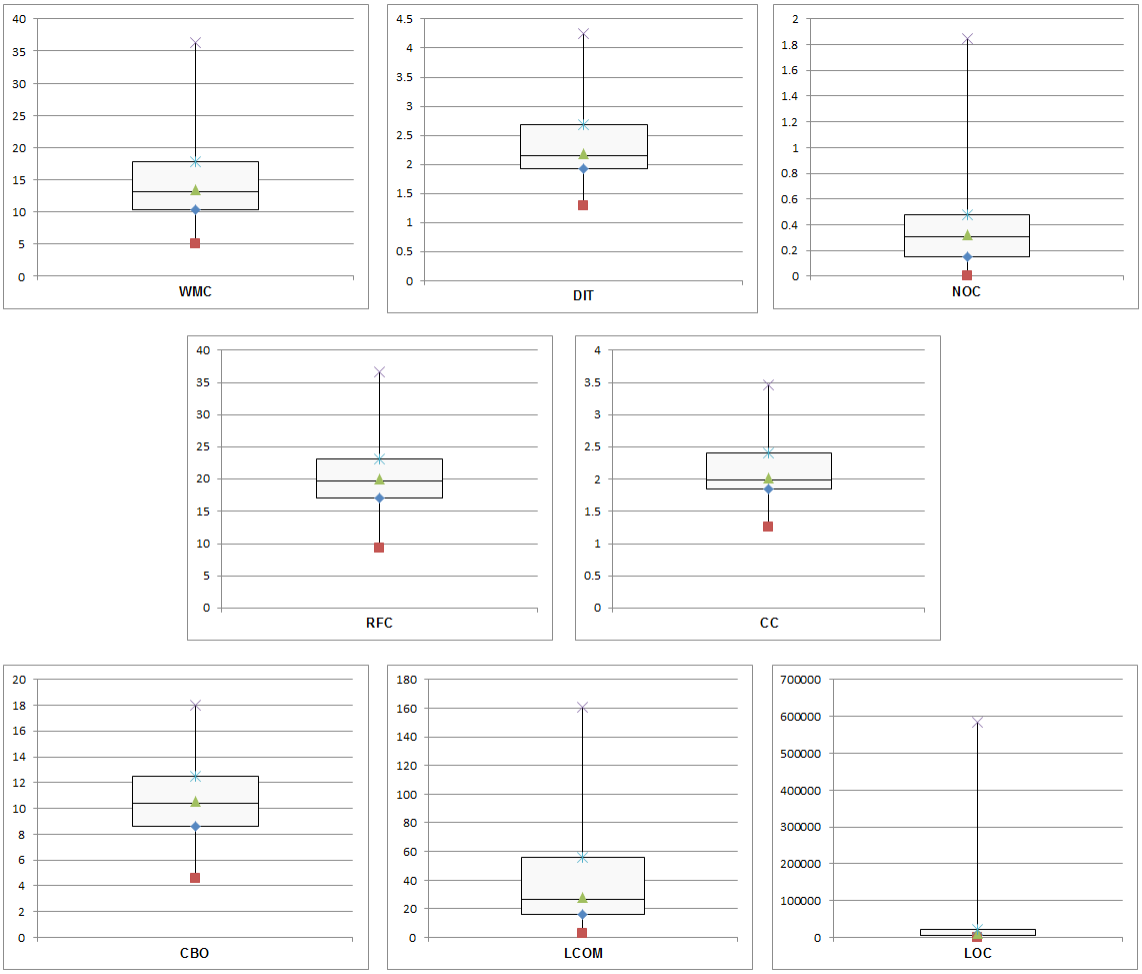


Figure 22: Data distribution of source code metrics

b) Descriptive analysis of market metrics

The second question of our GQM model pursues to understand what is the current performance of the mobile applications with respect to market success. We used the market success model based on the store metrics obtained for each application.

In a similar way, we present a summary of the most relevant descriptive statistics (Table 16) and the non-parametric data distribution (Figure 23) of the market metrics, which provides us with the necessary data to know the performance of the set of applications in the market, giving answer to question Q.2 of our GQM model.

Table 16. Descriptive statistics for market oriented metrics of 100 Google Play applications

	Penetration		Satisfaction
	NOD	NOR	AR
Minimum	500	9	3
1st Quartile	30,000	437	4.2
Median	300,000	1,247	4.4
Mean	2,458,384	14,648.41	4.3
3rd Quartile	1,875,000	6,883.50	4.6
Maximum	30,000,000	270,667	4.8
Standard Deviation	6,606,744.36	41,705.23	0.38

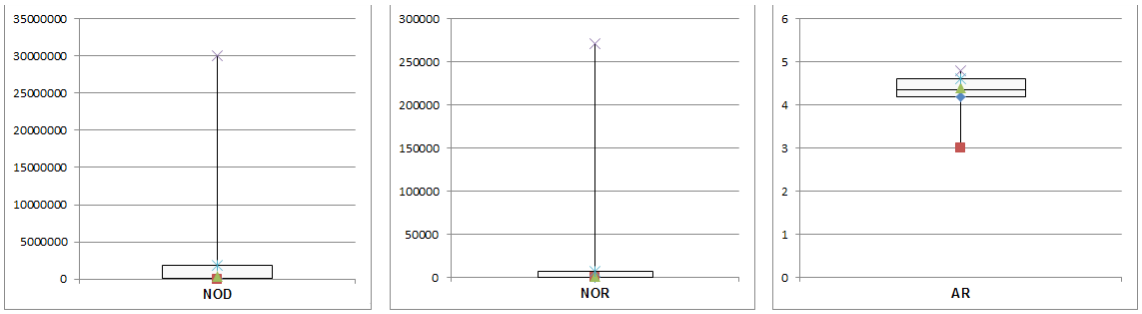


Figure 23: Data distribution of app market metrics

c) Simplification of the models

To prepare our data for the analysis, it is necessary to conduct an initial analysis to determine if two or more parameters show statistical dependence, that is, to determine if two or more metrics describe the same attribute. If such a case occurs, redundant variables can be omitted. To do it, it is necessary to calculate the correlation index among the analyzed variables, and highly correlated variables are assumed to describe the same attribute.

There are different methods to calculate correlation. Since the calculated metrics represent parameters measured in different scales, we utilized a non-parametric statistical method, which does not assume that the data have any characteristic structure. Moreover, the collected datasets for each metric do not necessarily followed a normal distribution. With these conditions in mind, we utilized the Spearman rank

correlation coefficient, which is an index (denoted as ρ) to calculate the correlation between two non-parametric variables, without assuming that the analyzed data are normally distributed. The Spearman rank correlation coefficient is also appropriate when one or both variables are skewed, and is more robust to incorporate outliers (Mukaka 2012). For the purpose of our analysis, the data is interpreted as follows:

- $0 < |\rho| \leq 0.25$: low correlation.
- $0.25 < |\rho| \leq 0.75$: moderate correlation.
- $|\rho| < 0.75$: high correlation.

c) Correlation of source code metrics

First, we calculated the Spearman non-parametric correlation index for the product oriented metrics (Table 17). We completed the analysis determining the level of significance of the correlations, indicating the *p* values of each correlation. In the table, the *p* values are expressed following the notation: $^{*}p < 0.05$, $^{**}p < 0.01$, $^{***}p < 0.001$. The r indices showed strong statistical dependency (closer to 1.0) between two pairs of metrics, namely WMC with CC, and WMC with RFC. Other metrics were moderately related, but we cannot suggest that they describe the same attribute since their r indices were not so close to 1.

Table 17. Non-parametric rank correlation (Spearman) of product oriented metrics

	Complexity				Coupling		Cohesion	Size
	WMC	DIT	NOC	CC	RFC	CBO	LCOM	LOC
WMC	-	-0.1379	-0.0815	0.7614^{***}	0.8378^{***}	0.3612^{***}	0.6523^{***}	0.2003^{*}
DIT	-0.1379	-	-0.1036	0.0292	0.0356	0.3584^{***}	-0.2743^{**}	-0.2945^{**}
NOC	-0.0815	-0.1036	-	-0.2377^{*}	-0.2383^{*}	-0.2594^{**}	0.2604^{**}	0.5455^{***}
CC	0.7614^{***}	0.0292	-0.2377^{*}	-	0.4821^{***}	0.3363^{***}	0.1522	0.1444
RFC	0.8378^{***}	0.0356	-0.2383^{*}	0.4821^{***}	-	0.6206^{***}	0.5430^{***}	-0.0419
CBO	0.3612^{***}	0.3584^{***}	-0.2594^{**}	0.3363^{***}	0.6206^{***}	-	0.1572	-0.1320
LCOM	0.6523^{***}	-0.2743^{**}	0.2604^{**}	0.1522	0.5430^{***}	0.1572	-	0.4244^{***}
LOC	0.2003^{*}	-0.2945^{**}	0.5455^{***}	0.1444	-0.0419	-0.1320	0.4244^{***}	-

The ρ indices of WMC with CC, and WMC with RFC as well as the level of significance of these correlations denoted by the *p* values, permits us infer that WMC, CC and RFC describe the same attribute and as consequence we may simplify our product quality model by dismissing CC and RFC.

d) Correlation of market metrics

The Spearman correlation indices of market oriented metrics are shown in Table 18. To determine the level of significance of the correlations, we indicate the *p* values of each one based on the notation: $*p < 0.05$, $**p < 0.01$, $***p < 0.001$. In this family of metrics, we found a very high correlation between the number of downloads (NOD) and the number of reviewers (NOR), that is, two market penetration metrics. Other metrics did not show a strong correlation among them.

Table 18. Non-parametric correlation (Spearman) of market oriented metrics

	Penetration		Satisfaction
	NOD	NOR	AR
NOD	-	0.9261^{***}	0.2623^{**}
NOR	0.9261^{***}	-	0.3277^{***}
AR	0.2623^{**}	0.3277^{***}	-

The correlation between NOD as NOR is not surprising, given that the number of persons rating an application is certainly related to the actual number of users who had downloaded the application. Based on this analysis, we may dismiss NOD as NOR clearly describes the same attribute with the advantage of being an actual value, not an approximation.

With the objective of describing both the market penetration and customer satisfaction of the product, the two remaining variables, NOR and AR were used to set up a value called "Success Index" (SI). The SI is calculated multiplying the number of reviewers (NOR) by the Application Rating (AR) (4).

$$SI = NOR * AR \tag{4}$$

In this way, the SI is a scalar whose value is proportional to customer ratings (satisfaction) and to number of reviewers (penetration). The higher the SI is, the more successful is the product in the app store. The SI will consider more successful applications that are highly downloaded and at the same time are rated positively, giving them the highest SI. Applications that are rated positively but are not heavily downloaded would not have a high SI; similarly, applications hat are highly downloaded but are poorly rated would not have a high SI value.

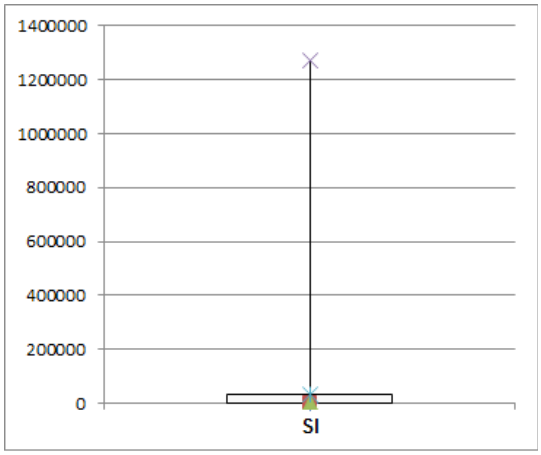
We calculated the SI for all the analyzed applications, and to have a high level representation of the new parameter, we derived the descriptive statistics, shown in Table 19. The non-parametric data distribution of the SI index is shown in the box plot of Figure 24.

The calculation of the SI provides the data required to provide an answer to our question Q3, which concerned the relationships between product quality and market success.

Table 19. Descriptive statistics for the success index of 100 Google Play applications

	Success
	SI
Minimum	41.00
1st Quartile	1878.00
Median	5237.00
3rd Quartile	30032.50
Maximum	1272135.00
Mean	65710.58
Standard. Deviation	189045.11

Figure 24: Data distribution of the success index metric



7.2.6 Understanding the impact of product quality in app store success

The third and last question of our GQM model utilizes the result of the previous two to determine if the source product quality contributes to the success of the mobile software product in the application store. According to this criterion, we consider as dependent variables those defined by the product quality model (i.e., the source code metrics), while the independent variable is the market success index calculated after the app penetration and rating values (i.e., the success index).

a) Hypothesis statement

With these considerations in mind, our next step is to look for any statistical dependence between the independent variables and the dependent variable, that is, any evidence of the influence of the source code metrics upon the market success index. To conduct the analysis, we used the taxonomy provided in the product quality model: complexity, coupling, cohesion and size. Based on them, we established our work hypothesis.

- *H1. Complex applications have higher market success.* Simple applications that perform only easy operations may be seen as to provide less services and to deliver less value to the final user. Therefore, applications that show higher complexity metrics might be expected to display a higher success index.

- *H2. Loosely coupled applications have higher market success.* Better coding style represents a better developed product that can perform better in user's hands. In this context, low coupling among classes is a fundamental design principle (Bavota 2013) aimed at reducing the system complexity (Marcus 2008). Tightly coupled systems show poor modularity, high module dependence, difficulties in reusability and overall problems on maintainablity. As a consequence, low values on coupling are expected to correlate to a higher success index.
- *H3. Highly cohesive applications have higher market success.* Similarly to H.2, this hypothesis deals with coding style quality. Provided that better style in code represents a better software implementation, high cohesion is a desirable property of software (Bavota 2013) as it positively impacts understanding, reuse, and maintenance (Marcus 2008). Consequently, high values on cohesion would correlate to a higher success index.
- *H4. Larger applications have higher market success.* As discussed in H.1, small applications are expected to perform less operation and in consequence to have less functionality. In user's hands they may represent less service, and less value. Therefore, applications of a larger size may be associated to a higher success index.

In short, our four work hypotheses assume that larger, complex applications that are developed with concerns about coding quality achieve higher market success.

b) Hypothesis testing

To test our work hypothesis, we calculated again the Spearman rank correlation coefficient, this time including the seven source code metrics of the simplified product quality model, and the market success index (SI), that represents the simplified market success model. The obtained correlation indices are shown in Table 20. To determine the level of significance of the correlations, we indicate the p values of each one based on our usual notation ($^{*}p < 0.05$, $^{**}p < 0.01$, $^{***}p < 0.001$).

Table 20. Non-parametric correlation (Spearman) between product oriented metrics and success index

	Complexity			Coupling	Cohesion	Size	Success
	WMC	DIT	NOC	CBO	LCOM	LOC	SI
WMC	-	-0.1379	-0.0815	0.3612^{***}	0.6523^{***}	0.2003^{*}	0.0357
DIT	-0.1379	-	-0.1036	0.3584^{***}	-0.2743^{**}	-0.2945^{**}	-0.2050
NOC	-0.0815	-0.1036	-	-0.2594^{**}	0.2604^{**}	0.5455^{***}	0.1178
CBO	0.3612^{***}	0.3584^{***}	-0.2594^{**}	-	0.1572	-0.1320	-0.1498
LCOM	0.6523^{***}	-0.2743^{**}	0.2604^{**}	0.1572	-	0.4244^{***}	0.1074
LOC	0.2003^{*}	-0.2945^{**}	0.5455^{***}	-0.1320	0.4244^{***}	-	0.3644
SI	0.0357	-0.2050	0.1178	-0.1498	0.1074	0.3644	-

We observe in Table 20 that the Spearman indices span in a range from -0.2050 to 0.3644, which do not describe a significant correlation according to our criteria. It is also interesting to note that the *p* values of the correlations do not show a high level of significance in the correlations values with the success index. In consequence, initially we cannot claim that the code quality attributes obtained from our product quality model correlate to the customer's perception on the final product and therefore its market success. However, to have a refined notion of the influence of each metric, we deepen the analysis studying how each group of quality attributes impact the success index separately. To achieve it, we implemented a series of regression models.

Regression analysis is a statistical tool to investigate relationships between variables, utilized to ascertain the causal effect of one variable upon another. In our case, using regression analysis we investigate the impact of different groups of source code metrics on the success achieved by the application in the marketplace. By conducting regression analysis, we build a model that assemble the independent variables at hand (code quality) and relate them statistically to a dependent variable (market success) through a linear function. When performing the regression analysis, it is also calculated the statistical significance of the estimated relationships, that is, the degree of confidence that the estimated relationship is close to an actual relationship. Depending on the number of independent variables that constitute the regression model we may

have simple regression, that pursues to describe the effect of one variable upon another, or multiple regression, that allows the incorporation of additional factors so that the effect of each one can be estimated separately (Skyes 2000) (Crawley 2005).

We conducted three simple regression models to refine our explanation of the relationship between coupling and market success, cohesion and market success and size and market success. Additionally, a multiple regression model was implemented to refine our description of the relationship between code complexity (described by three variables) and market success. Our regression model consists of the following values:

- *Intercept:* a constant that hypothesizes the value of the dependent variable with a zero value in the independent variable.
- *Estimate:* the effect of the independent variable upon the dependent variable.
- *Standard error:* the effect of additional factors that might influence the dependent variable.
- *t value:* Calculated statistic from the t-test.
- *p value:* The level of significance at which the hypothesis may be rejected.
- R^2 *value:* Indicates the percentage of the dependent variable that can be explained through the independent variable (the multiple R^2 is not dependent on the number of variables in the model, while the adjusted R^2 is).

The first three values (Intercept, Estimate and Standard Error) describe the linear relationship between the independent variables and the dependent variable. The *t* value, *p* value, and R^2 value describe the reliability of our regression model: as the *t* value arises, it denotes that the null hypothesis is false; the smaller the *p* value is, it means a lower probability to draw a wrong conclusion, and the closer to 1 the R^2 is, the more certainly the dependent variable can be described through the independent variables.

- *H1. Complex applications have higher market success.*

To explain the relationship between complexity and market success, we executed a multiple regression model. In Table 21 we observe a mixed interpretation among complexity metrics: WMC and NOC describe a positive relationship with respect to the success index; on the other hand, the relationship between DIT and SI is negative, which outlines that the less complex the app is, the more successful it is likely to be in the market. However, the only metric that might have a significant impact on the success index is DIT, achieving a *p* value less than 0.05, yet a high absolute *t* value. WMC and NOC shown *t* and *p* values that do not permit to consider these metrics significant for inference. Moreover, the R^2 value is close to 0, indicating that in general, the code complexity metrics fail in describing accurately the success of the application in the market. This analysis confirms the trend identified by the rank correlation values previously calculated, that suggested a poor relationship between complexity and market success. In consequence, this analysis supports the rejection of hypothesis H1.

Table 21. Multiple regression model: product complexity and market success index

	Success Index (SI)			
	Estimate	Std. Error	t value	Pr (> | t |)
Intercept	179355	118471	1.514	0.1334
WMC	2240	4262	0.526	0.6004
DIT	-72719	34349	-2.117	0.0369
NOC	101505	74579	1.361	0.1768
Multiple R^2	0.0715			
Adjusted R^2	0.0316			

- *H2. Loosely coupled applications have higher market success.*

The simple regression model effectively describes a negative relationship between the coupling metric and the success index (Table 22). Nonetheless, the results of the regression show that there is no significant impact of the CBO on the success index: the high *p* value endorses the poor probability of drawing a reliable conclusion, fact that is confirmed by the high absolute value of the *t* statistic. In addition, a very low value on R^2 indicates a poor level of certainty that

the independent variable helps in describing the dependent variable. The regression model endorses the trend observed in the Spearman correlation that described a very low relationship between complexity and market success. As conclusion of the regression model, we cannot claim that loosely coupled applications accomplish higher market success, and consequently we rejected the work hypothesis H2.

Table 22. Regression model: product coupling and market success index

	Success Index (SI)			
	Estimate	Std. Error	t value	Pr (> | t |)
Intercept	95599	74927	1.276	0.205
CBO	-2948	6807	-0.433	0.666
Multiple R^2	0.0019			
Adjusted R^2	-0.0084			

- *H3. Highly cohesive applications have higher market success.*

  A higher LCOM indicates lower cohesion. The simple regression model (Table 23) describes a slight positive relationship between LCOM and the success index. We expected a negative correlation since the lower are the values in LCOM, the higher values expected in SI. However, we cannot consider the results of the regression as clearly significant due to the values observed in the *t* and *p* statistics and the R^2 value. LCOM shows high *p* and *t* values, which denotes a high probability of drawing a wrong conclusion, and the resulting R^2 value is very close to zero, denoting the unfitness of the cohesion metric to describe the success index. The simple regression model confirms our observations in the low rank correlation index obtained, which shows no relationship between high cohesion and market success, endorsing the rejection of the work hypothesis H3.

Table 23. Regression model: product cohesion and market success index

	Success Index (SI)			
	Estimate	Std. Error	t value	Pr (> | t |)
Intercept	32907.0	30017.2	1.096	0.276
LCOM	809.5	599.3	1.351	0.180
Multiple R^2	0.0186			
Adjusted R^2	0.0084			

- *H4. Larger applications have higher market success.*

  The simple regression model effectively describes a very low (close to zero) positive relationship between the size metric (LOC) and the success index (SI) (Table 24). Additionally, the high *p* value denotes a very high probability to draw a wrong conclusion on the success metrics based only on the size metric. Finally, a very low R^2 value indicates a low level of certainty that the independent variable describes accurately the dependent variable. This trait was observed in the rank correlation exercise previously conducted, which outlined a very low relationship between lines of code and success index. As conclusion of the regression model, we cannot conclude that larger applications have a higher market success, and consequently we rejected the hypothesis H4.

Table 24. Regression model: product size and market success index

	Success Index (SI)			
	Estimate	Std. Error	t value	Pr (> | t |)
Intercept	63230	20560	3.075	0.002
LOC	0.046	0.338	0.139	0.889
Multiple R^2	0.0002			
Adjusted R^2	-0.0102			

7.2.7 Discussion

The values obtained after the different regression models lead to the rejection of the four work hypothesis that aimed to find a connection between the quality of the code and the market success on mobile applications. These conclusions are in line with the results of the initial correlation analysis, which described a very weak relationship

between the code quality metrics and the market success metrics. This permits us to state a substantiated answer to the third question of our GQM model: *"Does the source product quality contribute to the success of the mobile software product in the context of application markets?"* Utilizing a quality model that investigated code quality in terms of complexity, coupling, cohesion and size, and after several iterations of statistic analysis, it was shown that source code metrics have a minimal impact into the market success, described by a compound index that considers market penetration and customer satisfaction. As a conclusion, we cannot affirm that the quality of the source code effectively contributes to the success of the mobile software product in the context of Android open source applications deployed in the Google Play store.

In addition, our analysis showed in a consistent way that the selected source code metrics are not statistically reliable to describe the market success metrics. Thus, we can state that the source code metrics are not a trustworthy source to infer the quality of mobile software applications in the context of Android open source applications deployed in the Google Play store.

Generalizing, our survey confirms the marginal contribution of code quality in market success that was described in Chapter 5, where these quality attributes were noted as having a minimal contribution to market success, accounting, combined, for only 2.42% of the 100% of the accomplishment level, as shown in Table 25 which is a subset of Table 9.

Table 25. Mobile App Quality Model: code-relevant quality sub-characteristics

Quality Sub-characteristics (from ISO/IEC 25010)	Quality Characteristic Relative Weight
7.1 Modularity	0.55%
7.2 Reusability	0.55%
7.3 Analyzability	0.55%
7.4 Modifiability	0.55%
7.5 Testability	0.22%

Once we dismissed source code quality as a direct driver of the market success of a mobile application, a natural question is to find evidence that permits us to know what are the quality attributes that effectively drive the market success of a given application. To increase our knowledge, we reviewed several surveys that specialize on monitoring and analyzing the user's expectations in the mobile application market. This allowed us to identify with factual data what are the most important quality characteristics expected by regular customers of real-world application stores.

The survey conducted by Bowen (2012) points out product characteristics such as responsiveness and easiness of use as some of the most important to satisfy the user. Compuware (2012) mentions functionality, tolerance from errors, app speed, accessibility, responsiveness, performance, and user experience as the most important qualities to achieve market success. Keynote (2012) suggest speed and responsiveness. Finally, the academic survey conducted by Weeks (2013) points out speed, simplicity, and intuitiveness as the most market-relevant quality characteristics.

These trends were found not only in research projects and scientific surveys, but also in non-specialized literature[22,23,24]. Market success is widely seen as being a function of a number of attributes that go beyond software development techniques. Usefulness, provision of added value, visual engagement, originality, simplicity, etc. are considered the most important quality characteristics that a mobile product must meet to succeed in the market. Clearly, many of them cannot be achieved purely through the assurance of the application's source code.

The Mobile App Quality Model agrees with these sources, as it ranks functional appropriateness, non-repudiation, trust, pleasure, among similar others as the most relevant quality characteristics for a mobile application. It is important to remember that these characteristics were defined and ranked after analyzing the publishing requirements from different app stores (refer to Figure 11 and Figure 12 in Chapter 5).

[22] http://developer.vodafone.com/five-tips-successful-mobile-apps

[23] http://www.forbes.com/sites/avidlarizadeh/2013/07/19/eight-tips-for-a-successful-app

[24] http://www.businessinsider.com/digital-prunes-slingshot-app-2010-5

In general, the market-aware orientation of the Mobile App Quality Model is endorsed by the results of the referred surveys and additional literature; however, to draw definite conclusions on the matter, it is necessary to conduct a deeper study in the form of application analysis or market surveys.

7.3 Summary and conclusions

In this Chapter, we investigated the contribution of the code quality to the market success of mobile apps in the context of open source Android application promoted in Google play. We designed a product quality model and a market success model based on assorted quality characteristics; for product quality: complexity, coupling, cohesion and size; and for market success: penetration and satisfaction. We conducted a survey where we retrieved the source code of 100 applications and we calculated the metrics as per our models. Utilizing statistic methods like non-parametric correlation and linear regression we investigated if there is a relationship between product quality and market success; as result, it was consistently shown that source code metrics have a marginal impact on the indices that describe the market success. In consequence, we cannot affirm that the quality of the source code contributes effectively to the success of the mobile software product in the context of Android open source applications deployed in the Google Play store. These results confirm partially the organization of the Mobile App Quality Model, which disregards the contribution of code quality to the overall quality attributes that are expected on the mobile software product from an app market oriented perspective.

7.4 Chapter bibliography

(Barbagallo 2008) Barbagallo, D.; Francalenei, C.; & Merlo, F.; (2008) The impact of social networking on software design quality and development effort in open source projects. In *Proceedings of the 2008 International Conference on Information Systems.* pp. 201. Association for Information Systems.

(Bavota 2013) Bavota, G.; Dit, B.; Oliveto, R.; Di Penta, M.; Poshyvanyk, D.; & De Lucia, A.; (2013) An empirical study on the developers' perception of software coupling. In

Proceedings of the 2013 International Conference on Software Engineering (ICSE 2013). pp. 692-701. IEEE.

(Bowen 2012) Bowen, K.; & Pistilli, M.D.; (2012) *Student preferences for mobile app usage.* EDUCAUSE. Center for Applied Research. Available online: http://net.educause.edu/ir/library/pdf/ERB1210.pdf (Accessed on September 25th, 2013).

(Cambria 2013) Cambria, E.; Schuller, B.; Yunqing Xia; & Havasi, C.; (2013). New avenues in opinion mining and sentiment analysis. *IEEE Intelligent Systems.* vol. 28 (2) pp. 15-21. IEEE. doi:10.1109/MIS.2013.30

(Chidamber 1994) Chidamber, S.R.; & Kemerer, C.F.; (1994). A metrics suite for object oriented design. *IEEE Transactions on Software Engineering.* vol. 20 (6) pp. 476-493. IEEE. doi:10.1109/32.295895

(Crawley 2005) Crawley, M.; (2005). *Statistics: An introduction using R.* Wiley. ISBN: 0-040-02297-3. 2008.

(Compuware 2012) Compuware Corporation (2012) *Mobile apps: What consumers really need and want. A global study of consumer's expectations and experiences on mobile apps.* Available online: http://offers2.compuware.com/rs/compuware/images/Mobile_App_Survey_Report.pdf (Accessed on September 25th, 2013).

(Keynote 2012) Keynote Systems, Inc. (2012) *Mobile user survey: Mobile user preferences, habits and expectations.* Available online: http://www.keynote.com/docs/reports/Keynote-2012-Mobile-User-Survey.pdf (Accessed on September 25th, 2013).

(Marcus 2008) Marcus, A.; Poshyvanyk, D.; & Ferenc, R.; (2008). Using the conceptual cohesion of classes for fault prediction in object-oriented systems. *IEEE Transactions on Software Engineering.* vol. 34 (2) pp. 287-300. IEEE. doi:10.1109/TSE.2007.70768

(McCabe 1976) McCabe, T.J.; (1976). A complexity measure. *IEEE Transactions on Software Engineering.* vol. 2 (4) pp. 308-320. IEEE. doi:10.1109/TSE.1976.233837

(Meirelles 2010) Meirelles, P.; Santos, C.; Miranda, J.; Kon, F.; Terceiro, A.; & Chavez, C., (2010) A study of the relationships between source code metrics and attractiveness in Free software projects. In *Proceedings of the 2010 Brazilian Symposium on Software Engineering (SBES).* pp. 11-20. IEEE. doi:10.1109/SBES.2010.27

(Midha 2008) Midha, V.; (2008) Does complexity matter? The impact of change in structural complexity on software maintenance and new developers' contributions in open source software. In *Proceedings of the International Conference on Information Systems (ICIS 2008).* pp. 37.

(Mukaka 2012) Mukaka, M. M.; (2012). A guide to appropriate use of correlation coefficient in medical research. *Malawi Medical Journal.* vol. 24 (3) pp. 69-71.

(Nagappan 2005) Nagappan, N.; Williams, L.; Osborne, J.; Vouk, M.; & Abrahamsson, P.; (2005) Providing test quality feedback using static source code and automatic test suite metrics. In *Proceedings of the 16th IEEE International Symposium on Software Reliability Engineering, 2005 (ISSRE 2005).* pp. 10. IEEE Computer Society. doi:10.1109/ISSRE.2005.35

(Santos 2012) Santos, C.; Kuk, G.; Kon, F.; & Pearson, J.; (2012) The attraction of contributors in free and open source software projects, The Journal of Strategic Information Systems, vol. 22 (1), pp. 26-45, ISSN 0963-8687, doi:10.1016/j.jsis.2012.07.004

(Scotto 2004) Scotto M.; Sillitti A.; Succi G.; Vernazza T.: (2004). Dealing with software metrics collection and analysis: a relational approach. Studia Informatica Universalis vol. 3 (3) pp. 343-366. Suger.

(Scotto 2006) Scotto, M.; Sillitti, A.; Succi, G.; & Vernazza, T.; (2006). A non-invasive approach to product metrics collection. *Journal of System Architectures.* vol. 52 (11) pp. 668-675. Elsevier. doi:10.1016/j.sysarc.2006.06.010

(Sillitti 2003) Sillitti, A.; Janes, A.; Succi, G.; & Vernazza, T.; (2003) Collecting, integrating and analyzing software metrics and personal software process data. In *Proceedings of the 29th Euromicro Conference, 2003 (EUROMICRO 2003)* pp. 336-342. IEEE. doi:10.1109/EURMIC.2003.1231611

(Skyes 2000) Sykes, A. O. (2000). An Introduction to Regression Analysis. In *Chicago Lectures in Law and Economics,* Eric A. Posner (Ed.) University of Chicago. ISBN: 978-1566629720

(Stamelos 2002) Stamelos, I.; Angelis, L.; Oikonomou, A.; & Bleris, G. L.; (2002). Code quality analysis in open source software development. *Information Systems Journal.* vol. 12 (1) pp. 43-60. Wiley. doi:10.1046/j.1365-2575.2002.00117.x

(Succi 2005) Succi, G.; Pedrycz, W.; Djokic, S.; Zuliani, P.; & Russo, B.; (2005). An empirical exploration of the distributions of the Chidamber and Kemerer object-oriented metrics suite. *Empirical Software Engineering.* vol. 10 (1) pp. 81-104. Kluwer Academic Publishers. doi:10.1023/B:EMSE.0000048324.12188.a2

(Weeks 2013) Weeks, K; (2013) *Going mobile: A guide to mobile website and application user preferences.* California Polytechnic State University, USA Available online: http://digitalcommons.calpoly.edu/cgi/viewcontent.cgi?article=1074&context=grcsp (Accessed on September 25th, 2013).

Chapter 8:

Summary and Conclusions

In this thesis we aimed to contribute to the configuration of a software assurance model that, leveraging mature software product quality methodologies, identifies and accommodates the mobile-specific quality needs. The activities included in this thesis included theoretical investigation, practical experimentation and market-aware surveys.

We identified the most relevant quality needs of the mobile app ecosystem, based upon the publishing guidelines of the major mobile application stores. To define a unified assurance model that satisfies widely these quality needs, we leveraged the ISO/IEC 25010 software quality standard to outline the mobile-specific quality characteristics. As a result, we obtained the Mobile App Quality Model, outlining mobile-specific quality characteristics defined on top of a solid software product quality standard. Finally, we provided a GQM-inspired approach that assists in the definition of metrics for the quantitative evaluation of the software product. We exercised the Mobile App Quality Model through case studies that measured relevant quality characteristics of custom applications, and on a selection of mobile apps surveyed from a real-world app store.

The software quality instruments developed in this thesis are designed to provide a comprehensive, objective and quantitative way to satisfy the requirements of mobile applications from an environment-specific point of view. This may be useful for developers in the assurance of their mobile apps throughout the development cycle and after the introduction in the market. Moreover, these quality tools can assists

customers and end-users in the recommendation, audit and overall evaluation of the mobile software product.

8.1 Research questions revisited

H.1: The mobile software product can be analyzed to relate measurable parameters of the software product with user expectations and market compliance criteria, delivering an accurate approximation of the product quality.

With the introduction of the Mobile App Quality Model, we furnished an assurance model that describes the most important quality characteristics of mobile applications based on the quality requirements from major mobile application stores. We are able to describe in a standardized approach the expectations of the end user, the constraints of the execution environment, and the software quality policies required to enter into the mobile market. For the association between the quality characteristics of the assurance model and measurable quality parameters, we proposed a partially instantiated GQM, designed to facilitate the definition of product metrics. To describe in detail this answer, we revisited our two research questions individually.

RQ.1: What are the most relevant quality requirements set upon a mobile application?

Utilizing a market awareness approach, we reviewed the publishing guidelines of the major mobile app stores that serve the widest range of operating platforms. This analysis permitted us to identify the most relevant product quality requirements and to classify them according to an area of interest. Our analysis allowed us to point out traits of the quality requirements of mobile applications. From the development point of view, application markets focus on guaranteeing products without flaws, that do not lead to any malfunction in the target device. There are as well strong requirements regarding usability, attractiveness, aesthetics, and resource management.

Scientific literature and design practices recommend development practices, coding style guidelines and design standards, but they do not dwell much on the quality requirements of application stores. App stores require the final product to be attractive,

safe and flawless for the final user and the device regardless of the way it is internally organized and coded. Characteristics such as modularity, reusability, analyzability, modifiability, testability were classified as little relevant. On the other hand, requirements tend to focus on customer satisfaction, denoted directly by the trustworthiness, usefulness, comfort, effectiveness and efficiency of the mobile application; that is, the ability of the application to perform correctly and help the user to solve a need in a trustworthy, pleasant and gratifying way. These traits were later confirmed in a case study that analyzed assorted applications from a real market setting.

RQ.2: How can we measure the quality of a mobile software product?

To provide an answer, our approach was twofold: first, we investigated a family of relevant quality characteristics that serve as basis and foundations for a potential set of metrics. Then, we introduced a mechanism based on the Goal-Question-Metric approach to facilitate the definition of custom metrics.

For our first step, we leveraged the quality model of ISO/IEC 25010 to draw the relationship between the app store quality requirements previously identified with the quality characteristics defined in the standard. This relationship serves as a strategy to establish a quality characteristic for each quality requirements of the app store. Utilizing customer driven engineering, we associated the quality requirements from the analyzed app stores with the quality characteristics of the ISO/IEC 25010 standard, and we ranked the outcome of such association. We utilized the association and ranking to set up the Mobile App Quality Model: a mobile-specific, standard-based software quality model to appraise the quality of the mobile software product based on a target-specific point of view of real-world market requirements.

For the second step, to evaluate the quality of a product in the mobile domain, it is necessary to relate the mobile-specific characteristics with measurable attributes. We proposed to summarize the common requisites and prepare a GQM baseline that can be partially instantiated. This partially instantiated GQM works as a template customized depending on the scope of the desired quality assessment. The partially

instantiated GQM is furnished preliminarily with quality characteristics from the Mobile App Quality Model to facilitate the definition of product metrics required to keep track of mobile-specific quality drivers. Both the model and the GQM were exercised as part of the methodologies used in our three case studies.

8.2 Experiments, case studies and surveys

We deployed some case studies that let us understand the impact of several of the quality characteristics identified by the Mobile App Quality Model, and that allowed us to exercise the partially-instantiated GQM to define custom metrics.

In a first case study, we reviewed the performance of apps by analyzing the execution time of a task that utilizes specific software and hardware resources of the mobile device. We exercised our analysis using web-based and native programming. We obtained a set of data that allowed us to identify what are the resources that require more time to respond, and in what cases a web application performs better than a native application.

In a second case study, we analyzed the energy consumption of mobile apps by measuring the battery discharge of the target device after utilizing specific hardware resources. We measured the impact of the different components on the overall power consumption of mobile system. In this way, we identified the resources that are more energy-consuming and that as consequence may impact the design aspects or commercial success of the application.

In a third study, we assessed the source code quality attributes obtained from our product quality model, and after a survey of 100 Open Source mobile apps taken from a real application market we correlated the source code quality attributes with metrics that permitted us determine the customer's perception on the final product. Our survey confirmed the marginal contribution of code quality in market success, dismissed by the Mobile App Quality Model.

8.3 Limitations and threats to validity

Even though we conducted the research activities, surveys and experimentation at the maximum effort, we acknowledge that our work had to cope with limitations and other threats.

For instance, to exercise the Mobile App Quality Model we analyzed only applications deployed and marketed through the Android OS platform, neglecting other execution environments that may be characterized by a different profile. However, one should take into account that our analysis required the openness and unrestricted access that could be provided only by the Android OS architecture.

The survey of real applications had to be limited to Open Source apps, which represent only a subset of the reality of the mobile app market. Indeed, with the tools at hand it was not possible to analyze closed source applications from the product-oriented approach that our study required. To partially overcome this, we surveyed an extensive range of applications to cover a wide scope of developer and customer profiles.

8.4 Directions of future research

We highly recommend the conduction of replication studies that may contribute to overcome the limitations and threats to validity explained in the previous section. For instance, implementing the Mobile App Quality Model to analyze a set of applications deployed and marketed on a different operating system, like iOS or Windows Mobile.

The experiments and surveys reported in this thesis concentrated on a subset of the quality characteristics expressed in the Mobile App Quality Model. We advise the design and implementation of surveys and experiments that can furnish information about the evaluation of quality aspects of mobile apps that were not covered by our case studies.

An additional opportunity for further research is to incorporate comment-based sentiment analysis when analyzing app market metrics. Mobile application stores are a rich source of comments in which users directly provide feedback that helps to assess

the success of an app in the store. The textual feedback may complement the insights given by the numeric values that we utilized, like number of downloads, number of reviewers and application ratings.

8.5 Closing remarks

Mobile devices are currently the most important platform for the introduction and utilization of software products and services. The trends show a consistent growth in computing capabilities, number of users and distributed products, configuring a rich field of research of extraordinary potential and impact.

Software Engineering should not miss the opportunity of exploring how mobile software applications may overcome the challenges of the mobile environment, profit from the operational possibilities and business opportunities, and satisfy the needs of the end user. To accomplish this, it is necessary to assure that the mobile software product meets and exceeds the requirements and expectations of the end user, the mobile environment and the application market.

The results of this research work contribute to the State of the Art of Software Engineering shedding light for the accomplishment of high quality mobile software products, providing answers to the emerging and growing need of having mobile-specific software quality assurance processes and reference models.

APPENDIX A

DATASETS

Table A-1. Relationship matrix between demanded quality from app stores and ISO/IEC 25010 Internal/External Quality characteristics.

Table A-2. Relationship matrix between demanded quality from app stores and ISO/IEC 25010 Quality in Use characteristics.

Table A-3. Full raw data set of the metrics obtained from the PROM system and the Google Play market.

APPENDIX B

PUBLISHED PAPERS

2014

1. **Corral, L.;** Georgiev A.B.; Sillitti, A.; Succi, G.; (2014) Can execution time describe accurately the energy consumption of mobile apps? An experiment in Android. *Accepted for publication in the 3rd International Workshop on Green and Sustainable Software (GREENS 2014), in connection with ICSE 2014. Hyderabad, India.* ACM.
2. **Corral, L.;** Georgiev, A.B.; Sillitti, A.; Succi, G.; (2014) Method reallocation to reduce energy consumption: An implementation in Android OS. *Accepted for publication as a full paper in the 29th ACM Symposium on Applied Computing (SAC 2014).* ACM.

2013

3. **Corral, L.;** Sillitti, A.; & Succi, G.; (2013) Agile software development processes for mobile systems: Accomplishment, evidence and evolution. In *Proceedings of the 10th International Conference on Mobile Web Information Systems (MobiWIS 2013).* Lecture Notes in Computer Science, vol. 8093. pp. 90-106. Springer-Verlag Berlin / Heidelberg. ISBN: 978-3-642-40275-3. doi:10.1007/978-3-642-40276-0_8
4. **Corral, L.;** Sillitti, A.; & Succi, G.; (2013) Using a partially instantiated GQM to measure the quality of mobile apps. In *Proceedings of the 25th International Conference on Software Engineering and Knowledge Engineering (SEKE 2013).* pp. 520-524. Knowledge Systems Institute. ISBN: 978-1-891706-33-2.
5. **Corral, L.;** Georgiev A.B.; Sillitti, A.; & Succi, G.; (2013) A method for characterizing energy consumption in Android smartphones and tablets. In *Proceedings of the 2nd International Workshop on Green and Sustainable Software (GREENS 2013), in connection with ICSE 2013.* pp. 38-45. IEEE. ISBN: 978-1-4673-6267-2. doi:10.1109/GREENS.2013.6606420
6. **Corral, L.;** Sillitti, A.; & Succi, G.; (2013) Software development processes for mobile systems: Is Agile really taking over the business? In *Proceedings of the 1st International Workshop on Mobile-Enabled Systems (MOBS 2013), in connection with ICSE 2013.* pp. 19-24. IEEE. ISBN: 978-1-4673-6333-4. doi:10.1109/MOBS.2013.6614218

2012

7. **Corral, L.;** (2012) Standard-based strategy to assure the quality of the mobile software product. In *Proceedings of the 3rd annual conference on systems, programming, and applications: software for humanity (SPLASH 2012).* pp. 95-96. ACM. ISBN: 978-1-4503-1563-0. doi:10.1145/2384716.2384755

8. **Corral, L.;** (2012) Using software quality standards to assure the quality of the mobile software product. In *Proceedings of the 3rd annual conference on systems, programming, and applications: software for humanity (SPLASH 2012).* pp. 37-40. ACM. ISBN: 978-1-4503-1563-0. doi:10.1145/2384716.2384734
9. **Corral, L.;** Sillitti, A.; & Succi, G.; (2012) Mobile multiplatform development: An experiment for performance analysis. In *Proceedings of the 9th International Conference on Mobile Web Information Systems (MobiWIS 2012).* Procedia Computer Science, vol. 10. pp. 736-743. Elsevier. ISSN: 1877-0509. doi:10.1016/j.procs.2012.06.094
10. **Corral, L.;** Janes, A.; & Remencius, T; (2012) Potential advantages and disadvantages of multiplatform development frameworks – A vision on mobile environments. In *Proceedings of the 3rd International Workshop on Service Discovery and Composition in Ubiquitous and Pervasive Environments (SUPE 2012), in connection with MobiWIS 2012.* Procedia Computer Science, vol. 10. pp. 1202-1207. Elsevier. ISSN: 1877-0509. doi:10.1016/j.procs.2012.06.173
11. **Corral, L.;** Sillitti, A.; Succi, G.; Strumpflohner, J.; & Vlasenko, J; (2012) DroidSense: a mobile tool to analyze software development processes by measuring team proximity. In *Proceedings of the 50th international conference on Objects, Models, Components, Patterns (TOOLS 2012).* Lecture Notes in Computer Science, vol. 7304. pp. 17-33. Springer-Verlag Berlin / Heidelberg. ISBN: 978-3-642-30560-3. doi:10.1007/978-3-642-30561-0_3
12. **Corral, L.;** Janes, A.; Remencius, T., Strumplofhner, J.; Vlasenko, J.; (2012) A Novel Application of Open Source Technologies to Measure Agile Software Development Process. In *Proceedings of the 8th International Conference on Open Source Systems (OSS 2012).* Hammamet, Tunisia. IFIP AICT 378. ISBN: 978-3-642-33441-2. pp. 316-321. Springer-Verlag Berlin / Heidelberg. doi:10.1007/978-3-642-33442-9_28

2011

13. **Corral, L.;** Sillitti, A.; Succi, G.; Garibbo, A.; & Ramella, P; (2011) Evolution of mobile software development from platform-specific to web-based multiplatform paradigm. In *Proceedings of the 10th SIGPLAN symposium on new ideas, new paradigms, and reflections on programming and software (ONWARD 2011).* pp. 181-183. ACM. ISBN: 978-1-4503-0941-7. doi:10.1145/2048237.2157457
14. **Corral, L.;** Sillitti, A.; & Succi, G.; (2011) Preparing mobile software development processes to meet mission-critical requirements. In *Proceedings of the 2nd Annual Workshop on Software Engineering for Mobile Application Development, at with MOBICASE 2011.* pp. 9-11.
15. **Corral, L.;** Sillitti, A.; & Succi, G.; (2011) Managing TETRA channel communications in Android. In *Proceedings of the 13th International Conference of Enterprise Information Systems (ICEIS 2011).* vol. 3. pp. 307-312. SciTePress. ISBN: 978-989-8425-55-3.

APPENDIX C

PAPERS UNDER REVIEW

1. **Corral, L.**; Sillitti, A.; Succi, G.; Sarcià, S.A.; *Defining Relevant Software Quality Characteristics from Publishing Policies of Mobile App Stores.* Submitted to the 10th International Conference on Mobile Web and Information Systems (MobiWIS 2014). Submitted: 7/Mar/2014.

2. **Corral, L.**; Sillitti, A.; Succi, G.; *Relating Product Quality Attributes with Market Success in Mobile App Stores.* Submitted to the Springer Software Quality Journal, Special Issue on Software Quality for Mobile Apps. Submitted: 25/Oct/2013.

3. **Corral, L.**; Sillitti, A.; Succi, G.; *Software Assurance Practices for Mobile Applications.* Submitted by Invitation to the Springer Computing Journal, Special Edition on Mobile Web and Information Systems. Submitted: 5/Nov/2013.

Fakultät für Informatik
Facoltà di Scienze e Tecnologie informatiche
Faculty of Computer Science